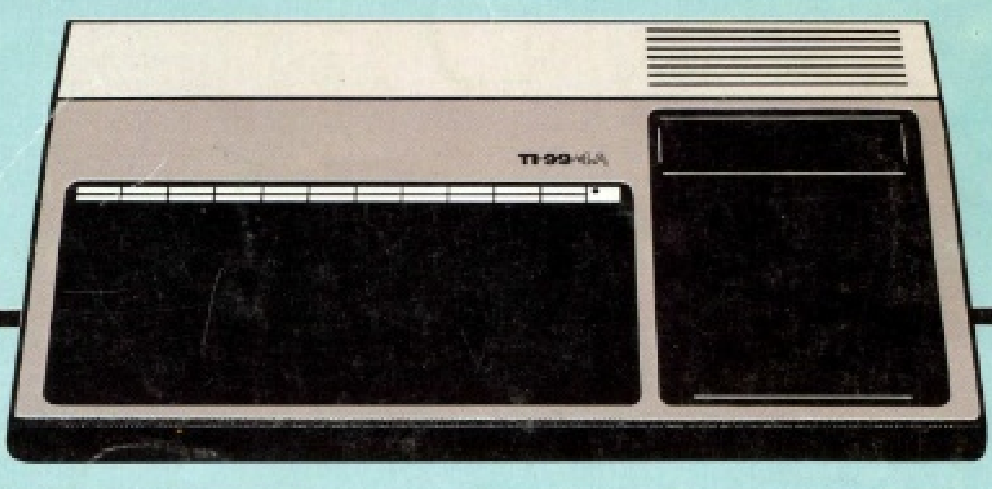
CASSETTE
D'ACCOMPAGNEMENT
DISPONIBLE

99 MAGAZINE

Complete Collection

DES ARTICLES PEDAGOGIQUES
DES BANCS D'ESSAI
DES PROGRAMMES
DES JEUX POUR VOTRE TI-99



NUMÉRO 1 - Trimestriel - Juin 1983 - 40 F

99 MAGAZINE

Sommaire

Page

Editorial par Hervé Thiriez	**5**
Music Maker à l'essai par Edouard Jouan	**6**
Pourquoi acheter un TI-99/4A par Alexandre Duback	**7**
A vous de programmer	**8**
Déballage du TI-99 par Didier Bicking	**9**
Les avantages du BASIC Etendu par Jean Sahal	**10**
Initiation au BASIC par Roger Didi	**13**
Jeu du Pendu par Edouard Jouan	**23**
Fabriquez votre joystick par Robert Peltier	**27**
Appelez moi KEY par Bernard Braun	**28**
Programmes d'aide à la décision par Didier Bicking	**31**
Jeu de l'Alphabet par Denise Amrouche	**33**
Le club d'utilisateurs TI-Education	**34**
Introduction au LOGO par Catherine Berdonneau	**35**
Les Tours de Hanoï par Denise Amrouche	**39**
Récupérez les programmes Apple par Hervé Thiriez	**40**
Le troisième menuet de Bach par Alexandre Duback	**42**
Le dernier robot par Edouard Jouan	**45**
Cherchez l'ancêtre par Robert Peltier	**51**
Gestion Personnelle de Fichiers par Didier Bicking	**54**
Quand le TI devient un harmonium par Gilbert Arribet	**57**
Fichiers en BASIC par Simon Tchek	**61**
Trucs et astuces	**65**

Les annonceurs

PSI : p. 2 – DURIEZ : p. 47 – LA REGLE A CALCUL : p. 50 – L'ORDINATEUR INDIVIDUEL : p. 67

Éditions MEV - 49, rue Lamartine - 78000 Versailles

(Dépot légal : 2e trimestre 1983)

DES CASSETTES

Tous les programmes publiés dans notre revue ont été réunis sur une cassette avant d'être listés. Dans le louable souci de vous épargner l'effort de les taper, nous avons recopié cette cassette et nous en commercialisons les copies au prix de 55 Frs TTC, fais d'envoi compris.

Pour le même prix unitaire, vous pourrez aussi acquérir deux cassettes contenant un autre choix de programmes :

CASSETTE 99A

MINEUR : un personnage hérite d'une mine d'or qu'il souhaite revendre. Pour cela, il doit prouver qu'elle contient bien du métal précieux en en rapportant une quantité minimale. Il va donc descendre dans cette mine, ceuser des galeries et tenter de trouver parmi les veines qui se présentent à lui celles qui sont aurifères.

Une multitude d'ennuis peuvent lui arriver : l'une des veines peut être en fait une nappe d'eau qui va inonder toutes les galeries d'un niveau inférieur à celui où il se trouve. Ou bien, elle est constituée de rochers durs à creuser, etc.

Ce programme est écrit en Basic simple et est donc accessible à partir de la configuration minimale.

PRET : le programme du calcul de prêt publié dans ce numéro dans l'article "Programmes d'aide à la décision" se limite à constituer un écran contenant les informations. Ceux qui possèdent une imprimante peuvent en obtenir plus. Le programme que nous publions ici réalise, sur imprimante, le tableau d'amortissement détaillé. Il permet ainsi de connaître pour chaque mensualité le montant des intérêts, de l'assurance et le capital restant dû.

Ce programme est écrit en Basic étendu et fonctionne avec une imprimante.

DIVISION : s'il n'y a aucune difficulté particulière à réaliser le quotient de A par B, il est en revanche plus compliqué d'obtenir un résultat comportant plusieurs centaines de décimales. Ce programme réalise cette opération, en vous permettant de choisir le nombre de décimales. Il est écrit en Basic simple et affiche ses résultats sur l'écran.

ELECTRONIQUE : quelle est la résistivité d'un ensemble de résistances montées en parallèle, ou, connaissant l'une des résistances, quelle doit être la valeur de la seconde pour que l'ensemble monté en série ait une valeur donnée ?

Ce programme, écrit en Basic simple, réalise ce genre d'opérations non seulement pour les résistances mais aussi pour les condensateurs et les inductances. Il est indispensable si, par exemple, vous voulez bricoler votre TI-99.

CASSETTE 99B

YAHTZEE : un magnifique programme, en provenance des Etats-Unis, que nous avons traduit en français. L'ordinateur jette les cinq dès, puis rejette deux fois de suite ceux que vous décidez de ne pas conserver. Lorsque vous avez réalisé l'une des figures désirées, vous déplacez une flèche pour faire marquer les points là où il le faut. L'illustration musicale est magnifique !

Ce programme est en Basic simple et ne nécessite donc que la configuration de base.

CHARDEF : un programme d'aide à la programmation qui nous vient de nos amis du club Tisoft-Home de Bruxelles. Vous créez un dessin à l'écran et il indique automatiquement les codes correspondant à ce dessin.

Ce programme est lui aussi en Basic simple.

REGRESSION LINEAIRE : ce programme, en anglais, vous demande combien de points vous voulez étudier puis les coordonnées de ces points. En réponse, il vous fournit l'équation de régression (F(x)=A+B★x), le coefficient de détermination, le coefficient de corrélation et l'écart moyen des estimations. Il demande ensuite une valeur de x et calcule la valeur correspondante de y.

Il s'agit là encore d'un programme en Basic simple, donc accessible avec seulement la console et le magnétophone à cassettes.

99 MAGAZINE

BON DE COMMANDE

Je désire recevoir la cassette

☐ 99 A : Mineur, Prêt, Division, Electronique 50 F TTC

☐ 99 B : Yahtzee, Chardef, Régression linéaire 50 F TTC

Nom ..

Adresse ..

...

Ces tarifs comprennent l'envoi postal en France Métropolitaine et CEE (voie aérienne exceptée)
Envoi par avion : nous consulter

Envoyez ce bon de commande et votre règlement à :

Editions MEV - 49 rue Lamartine - 78000 Versailles

Editorial

Nous sommes heureux de vous présenter le premier numéro de "99 magazine". Cela fait maintenant plusieurs mois que nous préparons le lancement de cette revue. Comme nous, vous avez certainement dû souffrir parfois de manquer de renseignements, vous avez cherché des idées de programmes, vous avez souhaité connaître l'utilité de tel ou tel module.

"99 magazine" est une revue d'utilisateurs du TI-99, dans laquelle sont publiés des programmes, des exemples d'utilisation, des articles pédagogiques, des bancs d'essai, ... Nous nous sommes fixé pour objectif de panacher dans chaque numéro les niveaux des articles, ainsi que les langages utilisés.

Les nombreux programmes que vous trouverez dans chaque numéro sont disponibles sur une cassette d'accompagnement vendue par correspondance. Elle est commercialisée exclusivement en vente directe, afin que son prix puisse être le plus bas possible. Vous connaissez le prix habituel de vente des programmes sur cassette ; comparez... Si nous vendions les cassettes dans les boutiques, leur prix serait sensiblement plus élevé; c'est pourquoi nous les vendons seulement par correspondance. Attention : il y a eu une erreur dans la publicité parue dans l'Ordinateur Individuel; nous vendons une cassette d'accompagnement, et non une disquette !

Afin de satisfaire votre grande soif de programmes, nous vous proposons aussi deux cassettes comportant chacune plusieurs programmes. Ces cassettes, baptisées "créativement" 99-A et 99-B sont le début d'une série qui s'enrichira à chaque numéro.

Nous vous prions d'avance de nous pardonner s'il y a des erreurs dans ce numéro : un porte-documents contenant toutes les corrections a été volé dans la voiture de Didier Bicking, et le peu de temps dont nous avons disposé pour les reconstituer et sortir le numéro 1 pour Micro-Expo n'a probablement pas été suffisant pour retrouver toutes les corrections prévues.

La qualité de "99" dépend de vous : nous attendons toutes vos contributions, qui peuvent être envoyées sur cassette ou disquette, avec le texte d'accompagnement sur papier et éventuellement en traitement de texte. De plus, n'hésitez pas à nous écrire pour nous faire part de votre impression et de vos suggestions pour "99". Si vous désirez une réponse, il est préférable de joindre une enveloppe timbrée ; cela nous facilite le travail et, partant, nous rend plus disponibles.

Trois contributions de taille occuperont sans doute une part importante de vos loisirs : tout d'abord celle de **Edouard Jouan** *qui a testé le module Music Maker. Il commente aussi le programme du jeu du Pendu qui nous a été gracieusement donné par le club Tisoft-Home de Bruxelles, ainsi que celui du dernier Robot, de même provenance.*

Didier Bicking, *collaborateur permanent de cette revue, a commencé par déballer son premier TI-99/4A et vous donne ses impressions à ce sujet. Il a ensuite testé le module de Gestion Personnelle de Fichiers et écrit deux programmes d'aide à la décision qu'il vous propose dans un troisième article.*

Robert Peltier *et son adorable épouse ont deux jumeaux très mignons mais un peu "fougueux". Pourtant, il souhaite les faire accéder à l'ordinateur tout en les tenant éloignés du clavier... Il a donc bricolé un joystick et vous en expose les principes et la technique. Il a aussi écrit un programme très original de jeu pédagogique. Le sujet traité dans ce programme est l'histoire de France, mais le même type de programme peut être utilisé pour n'importe quel autre sujet.*

Denise Amrouche *vous propose deux programmes de jeu de sa composition : le jeu de l'Alphabet et les tours de Hanoï.* **Catherine Berdonneau**, *spécialiste du LOGO, vous présente ce langage particulièrement destiné à faciliter l'accès de jeunes enfants aux techniques de la programmation.* **Jean Sahal** *détaille, quant à lui, les avantages du Basic Etendu.*

Roger Didi *vous apporte un très important article d'initiation au Basic tandis que* **Bernard Braun** *a épluché le fonctionnement de l'instruction CALL KEY.* **Alexandre Duback** *présente un programme musical en provenance de Bruxelles, mais, surtout, il vous explique (un peu tard peut-être !) pourquoi vous avez acheté un TI-99.*

Hervé Thiriez *vous aide à convertir pour le TI-99 des programmes rédigés pour Apple.* **Simon Tcheke** *vous montre comment procéder, en Basic, pour gérer des fichiers. Enfin,* **Gilbert Arribet**, *à l'aide d'un programme, transforme son TI-99 en harmonium. Il vous expose ce programme en plusieurs étapes.*

Hervé Thiriez

Rédacteur en Chef-Directeur de la Publication : Hervé Thiriez. **Rédacteur en Chef-Adjoint :** Didier Bicking. **Comité de Rédaction :** Thomas Coppens, Roger Didi. **Maquette :** Jacques Vincent. **Dessins :** Laurent Bidot. **Rédacteurs :** Denise Amrouche, Gilbert Arribet, Catherine Berdonneau, Bernard Braun, Alexandre Duback, Edouard Jouan, Robert Peltier, Jean Sahal, Simon Tcheke.

Régie Publicitaire : Force 7, Anne Jourdan, 39 rue de la Grande-aux-Belles, 75483 Paris Cedex 10. Tél. : (1) 238.66.10.
Diffusion auprès des boutiques et librairies : Editions du PSI, 44-51, rue Jacquard, BP 86, 77400 Lagny-sur-Marne. **Composition :** Télécompo, 13-15 avenue du Petit Parc, 94300 Vincennes. **Impression :** Imprimerie Rosay, 47, avenue de Paris, 94300 Vincennes. Tél. : (1) 328.18.63.

MUSIC MAKER à l'essai

Edouard Jouan

Le MUSIC-MAKER est un module commercialisé par Texas-Instruments au prix de 250 F. Il contient un programme dont le but est de vous assister dans la composition musicale.

La mise en marche du module est tout à fait étudiée pour mettre l'utilisateur en appétit : un morceau à deux voies sonores de J.S. Bach est joué alors qu'apparaît sur l'écran une portée avec des notes. Cela dure environ 30 secondes, après quoi l'ordinateur demande si vous voulez charger un fichier musical ou créer une nouvelle composition, soit par la méthode graphique, soit en mode traditionnel.

Dans ce dernier cas, il vous demande combien de dièses ou de bémol vous voulez disposer à la clé (ce qui permet de ne pas composer toujours en do majeur). Les signes de tonalité sont disposés dans l'ordre normal du solfège : fa do sol ré la mi si pour les dièses et si mi la ré sol do fa pour les bémols. Le compositeur choisit ensuite la mesure, mais seuls 1/4, 2/4, 3/4 et 4/4 sont possibles. Un air en mesure 9/8, par exemple, devra donc être réécrit pour pouvoir entrer dans ce cadre. Enfin, le compositeur choisit la vitesse d'exécution de la musique par une valeur de 1 à 30.

Pour composer une musique, cela est à la fois simple et compliqué : deux portées sont affichées à l'écran, une pour la clé de sol (les notes aiguës) et l'autre pour la clé de fa (les notes graves). A la droite de ces deux portées sont présentés les signes musicaux disponibles : ronde, blanche, noire, croche, double-croche, silence, demi-silence, soupir, demi-soupir, dièse et bémol.

En-dessous des signes, trois chiffres (1, 2 et 3) permettent de déterminer laquelle des trois voies devra être écrite. Trois autres chiffres permettent de faire jouer immédiatement ce que l'on vient d'écrire, pour l'une des voies seulement ou pour toutes celles qui ont été écrites. Un autre menu nous propose, selon le cas, soit de recopier ce qui a été écrit dans l'une des pages précédentes, soit d'effacer ce que l'on vient d'écrire pour l'une ou plusieurs des voies musicales.

Enfin, un dernier menu offre le choix suivant :

- PROC'D pour passer à la page suivante,
- REDO pour revenir à la page précédente,
- BACK pour revenir au menu de départ.

Dans les trois cas, la page qui figure à l'écran est mémorisée et il sera possible de la retrouver.

Comment procède le compositeur ? Un curseur est mis à sa disposition et il choisit en premier lieu celle des trois voies pour laquelle il va écrire. Il positionne le curseur sur le numéro de la voie et appuie sur ENTER. Il choisit ensuite le signe musical qu'il veut disposer dans la portée, toujours en lui superposant le curseur et en faisant ENTER.

Le signe musical choisi est alors représenté à la gauche de la partition, entre les deux portées. Le compositeur déplace ce signe à l'aide des quatre touches fléchées (E, S, D ou X), puis appuie sur ENTER. La note est alors jouée, ce qui permet en fait d'entendre la musique au fur et à mesure qu'on la compose.

La difficulté intervient lorsque la note suivante n'est pas représentée par le même signe, par exemple lorsque l'on vient de mettre une noire et qu'il faut maintenant une croche. Il faut alors déplacer le curseur sur le nouveau signe et faire ENTER : le signe choisi vient sur la portée en superposition du dernier signe écrit, mais sans effacer celui-ci. La même difficulté se présente lorsqu'il faut altérer une note, c'est-à-dire la munir d'un dièse ou d'un bémol. Il faut en effet écrire la note, aller chercher avec le curseur le signe d'altération, faire une fois ENTER pour amener ce signe sur la dernière note écrite et encore une fois ENTER pour enregistrer l'altération, puis ramener le curseur sur le prochain signe et refaire ENTER. Ce genre d'opération est très long et fastidieux.

Une fois la première voie écrite, le compositeur peut déplacer le curseur pour aller dans le premier menu indiquer qu'il va maintenant composer la deuxième ou la troisième voie. L'ordre de composition des voies est libre.

Un trait sympathique de ce programme réside dans une grille présentée au bas de l'écran. Cette grille se remplit au fur et à mesure que les notes sont disposées dans la portée. Il est ainsi possible de visualiser la durée de chaque note. De la sorte, le programme rend impossible l'écriture d'une note pour une voie si la note précédente de la même voie n'est pas terminée.

Chaque page présentée à l'écran représente une mesure. Le compositeur écrit alors cette mesure pour les trois voies, puis sélectionne PROC'D dans le dernier menu pour passer à la page suivante. Avant cela, il a la possibilité de faire jouer les notes qu'il voit à l'écran, individuellement pour chaque voie ou collectivement pour les trois voies. En revanche, si l'une des notes ne lui convient pas, il ne peut pas la modifier isolément : il doit effacer toutes les notes de la mesure dans la voie considérée, puis les réécrire en prenant bien garde cette fois de ne pas se tromper !

Le fichier musical ainsi créé peut bien entendu être sauvegardé, soit sur disque, soit sur cassette, pour être rappelé à une date ultérieure.

La composition en mode graphique se fait dans un système de coordonnées X,Y : le compositeur indique la durée de chaque note (celle-ci sera constante pour toute la composition : il faut donc choisir la durée la plus courte et la marquer plusieurs fois pour les notes longues), puis il place son pointeur à la hauteur souhaitée dans le graphe et fait ENTER. La composition à trois voies est possible ici aussi. En outre, un second graphe permet de faire évoluer séparément le volume de chacune des trois voies. Enfin, dans ce mode graphique, il est possible d'obtenir des quarts ou des dizièmes de ton : une option permet d'adopter un mode continu de sorte que la composition sera indépendante des fréquences prédéterminées pour les notes de la gamme classique.

Nous avons nous-mêmes essayé de créer avec ce module quelques compositions musicales. Dans un premier temps, nous avons écrit un canon à trois voies sur le thème de Frère Jacques. Il n'y a eu là aucun problème. En revanche, nous avons voulu transcrire des Inventions à deux ou trois voies de Bach et là, nous avons rencontré un obstacle insurmontable : Bach a malheureusement dépassé les possibilités du Music-Maker en écrivant des notes soit trop hautes, soit trop basses. Il aurait quand même pu y penser !

En fait les prétentions de ce module se réduisent bien à des compositions simples : les contraintes qu'il présente amoindrissent sensiblement les possibilités mêmes de l'ordinateur dans le domaine musical. Cependant, pour des compositeurs débutants, par exemple un élève de première année de Conservatoire, il illustre parfaitement bien le solfège qui devient tout à fait compréhensible. Les néophytes de la composition musicale trouveront dans le Music-Maker une assistance non négligeable.

Pouquoi acheter un TI-99/4A ?

Alexandre Duback

Par définition, 99 Magazine s'adresse à tous les possesseurs d'ordinateurs familiaux Texas Instruments ou compatibles. N'oublions cependant pas ceux qui, n'ayant pas encore plongé, se demandent "Pourquoi donc en achèterais-je un ?".

Cette interrogation dissimule en fait deux questions distinctes, auxquelles il est raisonnable de répondre séparément :

"Pourquoi un microordinateur familial"
"Le cas échéant, pourquoi un TI-99/4A ?"

L'intérêt d'un microordinateur familial.

De nombreuses raisons ont été avancées pour justifier l'achat d'un "micro" familial : familiarisation avec l'informatique, aspects ludiques (en bon français, les jeux...), gestion familiale, auto-formation, etc.

Familiarisation avec l'informatique

Je placerai en premier lieu le critère de familiarisation avec l'informatique : nous nous trouvons, qu'on le veuille ou non, dans une société déjà fortement informatisée, et qui le devient un peu plus chaque jour. Il est vital que les enfants apprennent, ne serait-ce que par osmose, ce qu'est un ordinateur, ses possibilités et ses limites.

D'ici quelques années, il ne se trouvera pas de bureau sans micro ou terminal d'ordinateur. D'ici une quinzaine d'années, de très nombreuses familles, dans les pays occidentaux, posséderont au moins un micro. La puce a déjà envahi l'électro-ménager ; elle pénètre actuellement dans les voitures. Peu d'appareils s'en passeront dans les années à venir !

Pour toutes ces raisons, il est important que nos enfants aient aujourd'hui l'occasion de se familiariser avec l'ordinateur. Quelle meilleure occasion pour cela que d'en avoir un à domicile ?

Aspects ludiques

Tout ce que font les jeux vidéo peut se faire, et en mieux, sur microordinateur. De plus, les micros permettent l'utilisation de jeux beaucoup plus variés et interactifs. Qu'il s'agisse de jeux à un ou à plusieurs joueurs, de jeux d'adresse ou de réflexion, de jeux d'aventures, tout est possible.

En outre, contrairement à ce qui se passe avec les jeux vidéo, on peut programmer ses propres jeux avec un micro familial. La participation au jeu peut ainsi revêtir un rôle nettement plus actif.

Gestion familiale

Il s'agit là un peu de l'argument "tarte à la crème". Il n'est pas si commode de mettre sa gestion familiale sur ordinateur : beaucoup de dépenses interviennent à des moments où l'on n'a ni le temps ni la motivation de noter le montant et l'objet de la dépense. Or, ces renseignements sont nécessaires pour un suivi détaillé du budget familial. De même, la mémorisation des recettes ou des menus est possible, mais souvent de peu d'intérêt pratique.

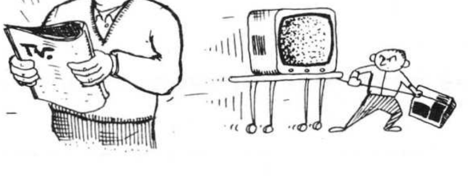
Par contre, le contrôle du compte en banque peut s'effectuer de façon nettement plus commode. L'utilisation d'un traitement de texte pour tout le courrier est bien pratique pour ceux qui écrivent souvent. Enfin, le recours à une "base de données" pour gérer un fichier trouve de nombreuses applications pratiques : gestion d'un agenda, fichier de disques ou de cassettes, bibliothèque, ...

Cependant, je ne considère pas les utilisations de l'ordinateur en gestion familiale comme étant une des meilleures raisons objectives d'achat.

Auto-formation

Voilà un domaine où les possibilités de l'ordinateur familial sont à peine exploitées aujourd'hui. L'ordinateur est un répétiteur d'une patience à toute épreuve, capable de s'adapter totalement au rythme de progression de l'élève, et sachant (s'il est bien programmé) individualiser la formation. Je me contenterai, pour en apporter la preuve, de raconter une histoire qui m'est arrivée.

Il y a deux ans, ma fille (neuf ans à l'époque) était totalement bloquée en mathématiques, incapable d'effectuer la moindre opération sans erreur.



Toute tentative de ma part pour lui faire faire des exercices le soir aboutissait en peu de temps à une crise de larmes.

En désespoir de cause, je lui ai écrit un petit programme, appelé PROF, qu'elle pouvait faire fonctionner toute seule. Ce programme lui permettait de choisir la nature d'une opération (addition, soustraction, multiplication ou division), puis lui fournissait une opération de la nature désirée, dont les chiffres étaient tirés au hasard.

Le programme mémorisait sur une disquette le nombre d'opérations de chaque type qu'elle avait traitées, et le nombre de fois où elle avait obtenu la bonne réponse à la première tentative, à la seconde ou pas du tout (elle avait droit à deux tentatives pour chaque problème). Je pouvais ainsi surveiller son travail a posteriori sans que ma présence ne soit nécessaire lors de la séance de travail.

Elle a fait de tels progrès en deux semaines que son professeur s'en est aperçu et en a fait la remarque. Pour que ce soit devenu visible dans une classe d'une trentaine d'élèves, il fallait que le progrès soit sensible !

Pour moi, cette expérience à elle seule a justifié après coup l'achat de mon ordinateur familial.

Développement de la créativité

Enfin, il ne faut pas négliger l'ordinateur comme outil d'aide au développement de la créativité. Il suffit pour s'en persuader de voir un groupe d'enfants développer des programmes en Logo.

Les possibilités de création graphique ou musicale à l'aide de l'ordinateur sont immenses. En outre, des outils tels que Logo n'isolent pas l'enfant dans son cocon, mais le font travailler en équipe ; et c'est un travail de création !

Pourquoi un TI-99/4A ?

De nombreux micros familiaux existent. Pourquoi choisir un TI-99/4A ? Probablement pour la première raison qui a fait la réussite d'Apple, dans une gamme de prix plus élevée : parce que c'est l'ordinateur familial qui offre la plus grande variété de programmes et de périphériques.

Texas Instruments représente actuellement 60% du marché français de l'informatique individuelle, et ce évidemment sans avoir bénéficié du concours explicite de l'Education Nationale ou d'autres organismes gouvernementaux. Si tant d'écoles et d'enseignants s'en sont équipés, c'est certainement le résultat d'une analyse de ce que pouvait apporter ce matériel.

Quels sont les points forts du TI-99/4A ?

- grande variété des programmes disponibles (plusieurs centaines)
- compatibilité avec de multiples périphériques et extensions
- micro-processeur 16 bits (Apple II ou Apple //e : 8 bits)
- écran haute résolution de 256*192 points (10% de plus que sur l'Apple II ou l'Apple //e)
- 16 couleurs
- majuscules et minuscules à l'écran
- possibilité d'extension de la mémoire à 48K
- clavier à touches mécaniques, plus fiables que les touches sensibles
- trois voies sonores d'origine, avec possibilité d'extension
- possibilité de synthèse et reconnaissance vocale
- langages disponibles : Basic, Extended Basic, UCSD Pascal, TI Logo, Assembleur-Editeur
- rapport performance/prix
- possibilité de partager son expérience avec des milliers d'autres utilisateurs
- et surtout, l'existence d'une excellente revue d'utilisateurs, n'est-ce pas ?

Apple l'a déjà démontré; il existe dans la micro-informatique un cercle vertueux (le contraire du cercle vicieux) reliant la diffusion d'un matériel et la richesse de ses possibilités. Plus un matériel est diffusé, plus il est rentable pour les auteurs de logiciel de concevoir des programmes qui lui sont destinés, plus il y a de programmes, plus il se vend de machines... et ainsi de suite. C'est la meilleure garantie de l'intérêt d'acheter un TI-99/4A.

A vous de programmer

Nous vous proposons, dans cette rubrique, une idée de programme, différente à chaque numéro. Ce n'est pas un concours ! Si vous réussissez à créer et faire fonctionner le programme suggéré, vous méritez un grand BRAVO, mais c'est tout ce que vous gagnez... Vous pouvez cependant nous envoyer votre solution. La meilleure solution reçue le 30 juillet au plus tard sera publiée et fera gagner à son auteur un abonnement d'un an.

L'informatique est un domaine où il n'y a jamais une seule et unique solution à un problème donné. Il est donc exclu de pouvoir publier toutes celles que vous allez trouver, alors... nous nous résolvons à n'en publier qu'une, mais cela vous permettra d'enrichir votre programmathèque.

1 - Etablissez un programme qui dessine sur l'écran la carte de votre pays. Les principaux fleuves sont représentés; les villes importantes sont figurées par un petit carré, les villes moins importantes par un point. Aucun nom ne figure dans cette carte.

2 - Le programme contient la liste des villes figurant sur la carte, avec leurs coordonnées sur l'écran (utilisez l'instruction DATA pour construire ce fichier).

3 - Le nom d'une ville est choisi au hasard et affiché dans un coin de l'écran. A l'aide du joystick ou des touches fléchées, le joueur doit amener un petit mobile sur le point ou le carré figurant la ville désignée.

4 - Après un certain nombre d'essais, le programme donne la solution au joueur si celui-ci ne l'a pas trouvée : il place lui-même le "mobile" sur le point ou le carré figurant la ville qui était demandée.

5 - Plusieurs villes sont ainsi demandées au joueur (à vous d'en déterminer le nombre), le nombre de réponses justes est cumulé et le pourcentage de bonnes réponses indiqué comme note au joueur en fin de partie.

Il vous appartient de déterminer les techniques et instructions à utiliser. Le jeu sera d'autant plus intéressant qu'il y aura un plus grand nombre de villes figurant sur la carte. Si vous vous sentez en verve, vous pouvez en faire autant pour les fleuves, mais c'est plus difficile. Une autre variante consisterait à écrire une version atteignant le même objectif avec les départements (ou cantons, ou provinces...) plutôt que les villes.

Pour ceux qui ne connaissent pas encore à fond leur cher TI-99, c'est là une excellente occasion de parfaire leur savoir...

99 magazine recrute

Vous recherchez une activité passionnante, tellement passionnante qu'elle occupera éventuellement une grande partie de votre temps.

Vous n'êtes pas trop pressé de percevoir d'importantes rémunérations mais pouvez en attendant vous contenter de peu.

Vous ne pouvez nous consacrer qu'un peu de temps, mais vous êtes intéressé par une activité comme la nôtre.

Venez vous joindre à notre équipe pour :

- ***rédiger ou analyser des articles,***
- ***rencontrer des auteurs,***
- ***rendre compte de salons ou d'expositions,***
- ***tester des matériels nouveaux,***
- ***analyser de nouveaux logiciels...***

Contactez-nous rapidement et nous nous rencontrerons pour convenir de modalités de collaboration.

Déballage du TI-99/4A

Didier Bicking

Notre premier travail, lors de la conception de cette revue, a évidemment consisté à étudier le TI-99. TEXAS-FRANCE a eu la gentillesse de mettre à cet effet un système absolument complet à ma disposition : celui-ci comporte la console, le boîtier d'expansion, le lecteur de disquettes et son contrôleur, une extension mémoire de 32K, et un choix de programmes dont notamment l'Assembleur.

J'ai, en premier lieu, été très impressionné par le nombre de cartons que cela représentait : pas moins de dix, de tailles variées, le plus gros étant celui du boîtier d'expansion. Mettre tout cela dans le coffre de ma voiture n'a pas été très difficile. Les problèmes sont apparus lorsque je suis arrivé chez moi : mon épouse, dont je salue ici la bonne composition, a quand même éprouvé de prime abord un certain ressentiment en assistant à ce débarquement.

Une fois tout déballé et monté, il ne restait finalement que peu de choses : la console, le boîtier d'expansion dans lequel avaient pris place toutes les cartes de périphériques ainsi que l'unité de disquette, et enfin les cassettes de programmes. Finalement, le plus dur aura été de caser quelque part les cartons vides avec leurs accessoires.

Le montage en lui-même ne présente pas de grande difficulté. Les cartes d'expansion, entièrement blindées, s'enfichent parfaitement bien. La seule mauvaise impression est donnée par cette énorme prise à brancher sur le côté droit de la console et son câble qui semble être en caoutchouc. Il est certainement possible d'améliorer la présentation de ce branchement, ainsi que son côté pratique. On peut d'ailleurs remarquer que, sur toutes les documentations diffusées par Texas Instruments, les photos représentant l'ordinateur dans sa configuration complète sont prises sous un angle tel que l'on ne voit ni la prise, ni le câble !

On comprend aisément le souci du constructeur de faire du solide : il s'agit là d'un matériel familial, donc destiné à être manipulé par des catégories de personnes très diverses. Sans doute le père de famille sera-t-il à cet égard beaucoup plus soigneux que son bambin de cinq ans qu'il commence à initier.

Tout cela est cependant un peu gros : le boîtier d'expansion est lourd et encombrant, son déplacement nécessite une certaine force. De plus, tout ce blindage et ces artifices de protection coûtent cher et arrivent finalement à empêcher, chez certains utilisateurs peu fortunés, un développement rapide du système. Si on compare avec d'autres systèmes équivalents, on remarque que les cartes sont ailleurs sans doute moins bien protégées, mais par contre plus économiques (à qualité égale) et moins encombrantes...

Une fois réalisé le montage du système lui-même, il ne m'a donc plus fallu que le brancher pour le mettre en route. Alors là, gros problème : mon téléviseur couleur, pourtant pas très vieux, ne comporte pas de prise Péritel... Ce problème est vraiment le plus monumental qui soit : alors que la console coûte aux environs de 2000 francs, il m'a fallu en ajouter 3500 de plus pour me permettre de la faire marcher. J'ai interrogé Texas Instrument à ce sujet : il semblerait finalement qu'une interface permettant de brancher l'ordinateur sur l'antenne soit en cours d'élaboration. Espérons-en la sortie pour bientôt ! Il serait en effet dommage que d'autres personnes rencontrent le même problème et doivent en passer par la même solution.

Enfin, dernier problème : caser le tout quelque part dans mon appartement sans abattre de cloison. Cela n'a pas été facile et je me suis finalement décidé pour la fabrication d'un meuble, pas trop laid je pense, dans lequel j'ai pu caser le tout : console, boîtier d'expansion et téléviseur, ainsi qu'une surface de rangement pour la documentation, parmi laquelle je prévois bien entendu les futurs numéros de "99 magazine".

Le premier contact ayant été établi, j'ai ensuite commencé à consulter la documentation du TI-Basic. Une bonne surprise m'attendait : cette documentation est en français; de plus, elle semble bien faite, clairement présentée et illustrée d'un nombre suffisant d'exemples. La lecture des quelques dizaines de pages contenant les principes du Basic est plutôt agréable, l'apprentissage se faisant avec une certaine progressivité. Mais la meilleure surprise survient dès la mise en marche du TI-99 : la mire en couleur qui se présente dès l'allumage laisse entrevoir les possibilités graphiques que nous allons pouvoir utiliser.

En garçon très méthodique, j'ai suivi les conseils donnés par la documentation, c'est-à-dire que j'ai commencé par allumer le boîtier d'expansion, puis la console et, enfin, le téléviseur. Il se passait quelque chose qui m'a semblé anormal : l'unité de disque interne (c'est-à-dire incluse dans le boîtier d'expansion) tournait en permanence, qu'elle soit ouverte ou fermée, qu'il y ait ou non une disquette dedans. Tout se passait en fait comme si les branchements entre cette unité et la carte du contrôleur avaient été faits à l'envers.

Pourtant, les prises de branchement du câble de liaison sont munies de détrompeurs devant en principe empêcher ce genre de fausse maneuvre. Il n'y avait donc pas d'erreur de ma part. J'ai alors pris le risque de briser ce détrompeur du coté de l'unité de disquette et de faire le branchement dans le sens inverse de celui qui semblait prévu... et cela a marché ! Une seule conclusion est possible : la prise avait été montée à l'envers. Il semblerait que le cas soit relativement fréquent, ou doive concerner une série de machines sorties pendant une durée limitée des chaînes de Texas Instrument ! Que ceux qui ont acquis le boîtier d'expansion avec l'unité de disquette ne s'affolent donc pas s'il leur arrive la même aventure.

Ce problème ayant été surmonté, il s'en est de suite présenté un autre : dans quel sens introduit-on la disquette dans le lecteur ? Il y a en effet huit manières différentes possibles, et la documentation est extrêmement discrète à ce sujet. La solution est, en fin de compte, assez logique : si l'on considère que le lecteur est disposé en dehors du boîtier, à plat à coté de celui-ci, le sens normal d'introduction de la disquette sera avec l'étiquette vers le haut et l'encoche à gauche. Lorsque l'on redresse le lecteur pour le mettre dans la position où il se trouve dans le boîtier, l'étiquette passe à droite, et l'encoche en haut. C'est effectivement dans ce sens là qu'il faut disposer les disquettes.

En fin de compte, je pense que dans l'ensemble, sauf en ce qui concerne la liaison entre la console et le boîtier d'expansion, l'ensemble est bien conçu pour une utilisation familiale. Tout parait solide et fiable, le clavier est d'un maniement agréable. Les lecteurs de cette revue trouveront dans nos pages quelques-unes des possibilités offertes par le TI-99 et que laissait déjà supposer la mire d'allumage.

Les avantages du BASIC Etendu

Jean Sahal

On a coutume d'opposer les littéraires aux scientifiques; c'est une dichotomie commode lorsqu'il s'agit de comparer l'attitude des uns et des autres face à l'ordinateur. Ce dernier présente de nombreux attraits aux yeux des scientifiques pour des raisons que l'on sait, alors qu'il suscite quelques réticences de la part des littéraires. Ces derniers n'ont a priori pas d'idée bien précise du parti qu'ils pourraient tirer de l'ordinateur dans le cadre d'une discipline d'où les mathématiques sont exclues.

Il est vrai que l'ordinateur a été conçu à l'origine pour effectuer des calculs numériques. Mais on a eu vite fait de découvrir qu'il pouvait aussi bien, comme c'est le cas du TI-99, traiter des données de nature aussi variées que des chaînes de caractères, des images, la parole, etc... Dès lors, ses possibilités d'application se sont considérablement étendues. Pourtant, certaines idées préconçues font que l'ordinateur rencontre une certaine résistance à se faire accepter par les non scientifiques. Mais ces idées commencent à évoluer. Ainsi, par exemple, au club TI de Vélizy (destiné aux membres du corps enseignant), on rencontre des enseignants appartenant à des disciplines fort diverses : il y a par exemple une maîtresse d'école maternelle qui écrit des programmes animant de petits animaux pour ses écoliers, ou encore un grammairien qui construit un programme de conjugaison, etc...

Le présent article a pour objet de montrer que le TI-99 peut très bien être ignoré en tant que "centre de calcul", et avoir néanmoins des applications dans beaucoup de domaines. Pour cela, il convient de faire un tour d'horizon des principales instructions qu'il met à la disposition des utilisateurs, en particulier lorsque ce dernier n'est pas un scientifique et lorsqu'il programme en Basic Etendu. Pour la clarté du texte, nous ne ferons ici qu'une présentation de chaque instruction. Vous trouverez ces mêmes instructions, présentées de manère plus détaillée, dans un encadré à la fin de cet article et mises "en situation" dans le programme ci-après.

Chaînes de caractères

Pour le traitement des chaînes de caractères et pour tirer parti de l'interactivité, on dispose des instructions habituelles à tous les micro-ordinateurs. Soulignons toutefois l'instruction *ACCEPT AT* qui apporte une solution rapide et esthétique à tous les problèmes de mise en page (tout comme *DISPLAY AT* d'ailleurs) et qui laisse la faculté de filtrer les réponses du manipulateur au moyen de son complément *VALIDATE*. Cette dernière instruction peut se révéler très utile en pédagogie car c'est un excellent moyen de mettre le manipulateur sur la bonne voie : les bips graves et aigus qui y sont produits permettent de différencier les propositions adéquates de celles qui sont inacceptables.

Effets sonores

Pour les effets sonores, *CALL SOUND*, en permettant de jouer jusqu'à trois notes simultanément, autorise la création d'une mélodie avec son accompagnement. Cela produit un type de musique un peu particulier, mais du reste fort agréable. Une phrase musicale est souvent bienvenue dans un programme d'apprentissage pour marquer le passage d'une séquence à une autre ou pour souligner une bonne ou une mauvaise prestation. Le synthétiseur vocal, associé à *CALL SAY* ou *CALL SPGET*, se révèle très utile dans les programmes destinés à ceux qui comprennent l'anglais. Par exemple, une mauvaise prestation est bien mise en relief lorsqu'elle est accompagnée par les paroles "this is incorrect, try again". On peut remarquer au passage que l'intonation de "TRY AGAIN" est empreinte d'une certaine bienveillance, comme pour encourager celui que l'ordinateur oblige à revoir la partie de programme mal assimilée.

Effets visuels

Pour les effets visuels et l'art pictural, on dispose d'un choix d'instructions et de sous-programmes résidents très complet. Certaines instructions peuvent paraître complexes, mais elles font gagner pas mal de temps dans la réalisation de graphismes fixes ou animés assez élaborés. Il faut reconnaître qu'une certaine expérience est nécessaire pour découvrir tout le parti que l'on peut tirer d'instructions en apparance insolites comme *CALL SPRITE, CALL MAGNIFY, CALL PATTERN, CALL MOTION.* Le manuel d'utilisation du Basic étendu donne une illustration amusante de *CALL PATTERN.* On y explique comment, avec un minimum de lignes de programme, on peut réaliser une roue avec moyeux et rayons, comment la faire tourner et avancer en même temps sur une surface, ce qui est du plus surprenant effet.

Programmation de jeux

Enfin, pour ceux qui désirent réaliser leurs propres jeux vidéo ou autres programmes de topologie animée, on dispose, en plus des instructions dont nous venons de parler, de quelques autres qui permettent d'agir directement et instantanément sur les graphismes ou sur certains objets qu'on tient à manoeuvrer individuellement. On connait l'instruction *CALL JOYST* qui fait intervenir les manettes de jeu, mais il en existe de plus subtiles comme *CALL COINC* en particulier, qui est un moyen de traiter l'information donnée par la rencontre de deux mobiles (sprites). Il serait vain de décrire en détail les utilisations potentielles de telle ou telle instruction : elles sont presque illimitées et dépendent en grande partie de l'imagination de chacun et du contexte dans lequel on les fait intervenir.

Applications

Lorsque l'on est en mesure de tirer parti de toutes les ressources du TI-99 et du Basic étendu (c'est-à-dire après une période plus ou moins longue d'essais et erreurs, et peut-être aussi d'échanges d'idées et de contacts avec d'autres utilisateurs), on parvient à réaliser des choses absolument extraordinaires. On peut par exemple créer des jeux vidéo dont la complexité et les qualités graphiques ou sonores sont dignes de rivaliser avec celles des jeux produits par des professionnels et que l'on trouve dans le commerce. On pourrait aussi imaginer de fabriquer un cours d'anglais audiovisuel à partir de dessins animés, parlant, personnalisé, sur mesure, avec contrôle des acquisitions et notation de l'élève.

Comme on le voit, le TI-99 est assez éclectique : il y en a pour tous les goûts et toutes les spécialités. On

peut néanmoins se demander si, à vouloir réaliser ses programmes soi-même, on n'est pas amené subrepticement à exercer un deuxième métier qui finit par être envahissant, et si, au fond, il ne vaudrait pas mieux laisser tout cela à des professionnels. L'un n'empêche cependant pas l'autre : il existe un certain nombre de logiciels professionnels pour le TI-99, mais il serait dommage de se priver du plaisir de programmer soi-même.

Ce qui nous parait discutable dans le cas de l'ordinateur familial, c'est de vouloir à tout prix privilégier son aspect outil de travail. Cet aspect a certes son importance, mais le TI-99 présente également un aspect divertissement qu'il ne faudrait pas faire passer au second plan. Nous ne voulons pas parler ici des jeux de réflexe ou autres que l'on trouve tout faits dans le commerce. Nous pensons plutôt au divertissement de haut niveau intellectuel que procure l'activité de recherche ou de création associée à un certain type de programmation. Réaliser un programme est en quelque sorte un jeu.

Il y a donc plaisir dès que l'on a la possibilité de se prouver à soi-même ou aux autres que l'on existe et que l'on possède une certaine originalité. Toute activité créatrice, toute activité à travers laquelle on s'exprime peut devenir un jeu. Lorsque l'on demande à un groupe d'élèves de réaliser un même programme, on constate que l'itinéraire emprunté pour arriver au même but diffère de l'un à l'autre. Chacun a son style de programmation. "Fais voir ton programme et je te dirai qui tu es". Pour un même programme de musique, certains utiliseront *DATA* et *READ ;* d'autres écriront une ligne de programme pour chaque temps, et choisiront de composer leur programme en utilisant REDO pour changer l'accord et le numéro de ligne. Ceci montre bien que programmer est une certaine façon de s'exprimer.

Cela est valable aussi bien pour un programme de quelques lignes que pour un programme long et complexe destiné à créer un jeu vidéo ou un spectacle audiovisuel par exemple.

Dans tous les cas, et si l'on fait passer au second plan l'aspect utilitaire, on peut réaliser un jeu avec un aspect magique et merveilleux parce que tout se passe sur un écran de télévision. Voilà pourquoi la programmation peut devenir une passion comparable à celle du bridge ou des échecs. On n'en épuise jamais toutes les ressources et, quelle que soit la richesse ou le degré de perfection d'un programme, on ne se lasse jamais d'y revenir pour le modifier et y apporter des améliorations.
Le programme qui fait suite au présent article ne présente pas en lui-même un très grand intérêt. Il est là essentiellement pour illustrer les emplois possibles des différents types d'instructions citées. Je ne saurais trop vous encourager à vous reporter au manuel du Basic étendu pour analyser la structure de chacune d'entre elles. Le programme vous les présente "en situation".

Voir encadré page 38

```
10 REM ====================
20 REM COPYRIGHT 99 MAGAZINE
30 REM ET JEAN SAHAL
40 REM ====================
50 REM
60 DISPLAY AT(4,4):"CECI EST UNE DEMONST
RATION DES POSSIBILITES DU TI 99 DANS DE
S DOMAINES QUI N'ONT RIEN DE SCIENTIFIQU
ES"
70 INPUT "PRESSEZ ENTER SI VOUS VOULEZ C
ONTINUER":Z$
80 CALL CLEAR
90 CALL SCREEN(10)
100 DISPLAY AT(4,8):"      POUR CONNAITRE
 MES TALENTS IL FAUT D'ABORD DEVINER LE
MOT DE PASSE QUI EST UNE VILLE DES USA B
IEN CONNUE"
110 PRINT "PRESSEZ LES LETTRES PUIS  ENT
ER"
120 ACCEPT AT(9,2)BEEP SIZE(7)VALIDATE("
DALS"):A$
130 IF A$="DALLAS" THEN 140 ELSE 120
140 CALL CLEAR
150 PRINT "JE PEUX PARLER MAIS SEULEMENT
 EN ANGLAIS" :: CALL SAY("HELLO,I+AM+A+#
TEXAS INSTRUMENTS#+COMPUTER")
160 PRINT
170 PRINT
180 PRINT "JE PEUX AUSSI JOUER DE LA MUS
IQUE PAS SEULEMENT LA MELODIE, L'ACCOMPA
GNEMENT AUSSI"
190 FOR A=1 TO 2
200 D=300
210 V=1
220 CALL SOUND(D,131,V,165,V,330,V)
230 CALL SOUND(D,392,V)
240 CALL SOUND(D,131,V,175,V,440,V)
250 CALL SOUND(D,131,V,196,V,523,V)
260 CALL SOUND(D,494,V)
270 CALL SOUND(D,131,V,175,V,440,V)
280 CALL SOUND(D,147,V,196,V,392,V)
290 CALL SOUND(D,147,V,196,V,294,V)
300 CALL SOUND(D,294,V)
310 CALL SOUND(D,131,V,175,V,262,V)
320 CALL SOUND(D*2,131,V,196,V,330,V)
330 NEXT A
340 PRINT
350 PRINT
360 PRINT "JE SAIS AUSSI FAIRE D'AUTRES
CHOSES " :: PRINT :: PRINT :: PRINT "VOI
CI DEUX EXEMPLES,CHOISISSEZ"
370 FOR B=1 TO 1000 :: NEXT B
380 CALL CLEAR
390 CALL SCREEN(12)
400 DISPLAY AT(8,2):"1 PETITE LECON D'AN
GLAIS"
410 DISPLAY AT(12,2):"2 DESSIN ANIME"
420 INPUT "TAPEZ LE CHIFFRE CHOISI PUIS
PRESSEZ ENTER: ":CHOIX
430 ON CHOIX GOSUB 440,940
440 CALL CLEAR :: CALL SCREEN(10):: PRIN
T "REGARDEZ LE CLAVIER,A GAUCHE VOUS VER
REZ DES FLECHES."
450 PRINT "SI VOUS APPUYEZ SUR CES TOUCH
ES VOUS DEPLACEREZ LE BALLON DANS LA DIR
ECTION CORRESPONDANTE"
```

```
460 PRINT "LES AUTRES TOUCHES(Q,W,T,A,F,
G,Z,C,V)ARRETENT LE BALLON,LA TOUCHE R A
RRETE L'EXERCICE"
470 PRINT
480 PRINT "============================"
490 PRINT :: PRINT "N'OUBLIEZ PAS:  EMPL
OYEZ GO QUAND VOUS PARLEZ DE QEULQUE CHO
SE QUI S'ELOIGNE,COME QUAND CA SE RAPPRO
CHE" :: PRINT
500 PRINT "============================"
510 PRINT :: PRINT "PRESSEZ ENTER SI VOU
S ETES PRET CAR A PARTIR DE MAINTENANT T
OUT SE FERA EN ANGLAIS"
520 INPUT REP$
530 CALL CLEAR
550 CALL SPRITE(#1,48,5,96,128)
560 CALL KEY(1,K,S)
570 IF S=0 THEN 560
580 IF K=5 THEN Y=-4 :: CALL CLEAR :: CA
LL SAY("UP "):: DISPLAY AT(6,14):"UP" ::
 PRINT "THE BALLOON IS GOING UP" :: GOSU
B 690
590 IF K=0 THEN Y=4 :: CALL CLEAR :: CAL
L SAY("DOWN"):: DISPLAY AT(6,14):"DOWN"
:: PRINT "the balloon is coming DOWN" ::
 GOSUB 810
600 IF K=2 THEN X=-4 :: CALL CLEAR :: DI
SPLAY AT(6,1):"COME TO THE LEFT!" :: CAL
L SAY("LEFT"):: PRINT "THE BALLOON IS CO
MING BACK"
610 IF K=3 THEN X=4 :: CALL CLEAR :: DIS
PLAY AT(6,18):"GO TO THE RIGHT!" :: CALL
 SAY("RIGHT"):: PRINT "THE BALLOON IS GO
ING AWAY"
620 IF K=1 THEN X,Y=0 :: CALL CLEAR :: P
RINT TAB(14);"STOP" :: PRINT :: PRINT "T
HE BALLOON IS STILL" :: CALL SAY("STOP")
630 IF K=6 THEN 670
640 IF K>6 THEN X,Y=0 :: CALL CLEAR :: C
ALL SAY("STOP"):: PRINT TAB(14);"STOP" :
: PRINT "THE BALLON IS MOTIONLESS"
650 CALL MOTION(#1,Y,X)
660 GOTO 560
670 CALL DELSPRITE(ALL):: CALL CLEAR ::
GOTO 380
680 CALL CLEAR
690 CALL CLEAR :: CALL MOTION(#1,-4,X)
700 CALL COLOR(4,4,4)
710 CALL HCHAR(1,1,61,640)
720 PRINT "WHAT IS THE BALLOON DOING ?"
730 ACCEPT VALIDATE("ITSGONUP ")BEEP:R$
740 IF R$<>"IT IS GOING UP" THEN 770
750 CALL SAY("#THAT IS RIGHT#"):: PRINT
"THAT IS RIGHT" :: PRINT :: PRINT "PRESS
 ANOTHER KEY"
760 GOTO 800
770 CALL SAY("#THAT IS INCORRECT##TRY AG
AIN#"):: PRINT "THAT IS INCORRECT" :: CA
LL CLEAR :: PRINT "CHOOSE BETWEEN GO UP
AND COME UP"
780 FOR DEL=1 TO 500 :: NEXT DEL
790 GOTO 680
800 RETURN
810 CALL CLEAR
820 CALL MOTION(#1,4,X)
830 CALL COLOR(4,4,4)
840 CALL HCHAR(1,1,61,640)
850 PRINT "WHAT IS THE BALLOON DOING ?"
860 ACCEPT VALIDATE("ITSGONUPDW CM")BEEP
:R$
870 IF R$<>"IT IS COMING DOWN" THEN 890
880 CALL SAY("#THAT IS RIGHT#"):: PRINT
"THAT IS RIGHT" :: PRINT :: PRINT "PRESS
 ANOTHER KEY" :: GOTO 910
890 CALL SAY("#THAT IS INCORRECT##TRY AG
AIN#"):: PRINT "THAT IS INCORRECT" :: CA
LL CLEAR
900 PRINT "CHOOSE BETWEEN GO DOWN  AND
COME DOWN" :: FOR DEL=1 TO 500 :: NEXT D
EL :: CALL CLEAR :: GOTO 820
910 RETURN
920 BREAK
930 GOTO 380
940 REM DESSIN ANIME
950 CALL CLEAR
960 CALL COLOR(12,16,16)
970 FOR A=19 TO 24
980 CALL HCHAR(A,1,120,32)
990 NEXT A
1000 A$="0107182121414141FFFF414121211907O
080E09884848282FFFF8282848498E000"
1010 B$="01061820305C4681814246242C18070
0806018342462428181623A000C0418E000"
1020 C$="0106182C244642818146 5C302018070
0806018040C3A62818142622 43418E000"
1030 CALL CHAR(96,A$)
1040 CALL CHAR(100,B$)
1050 CALL CHAR(104,C$)
1060 CALL SPRITE(#1,96,5,130,1,0,8)
1070 CALL SPRITE(#2,96,5,130,1,0,-8)
1080 CALL MAGNIFY(3)
1090 FOR A=96 TO 104 STEP 4
1100 CALL COINC(#1,#2,2,C)
1110 IF C=-1 THEN 1170
1120 CALL PATTERN(#1,A)
1130 CALL PATTERN(#2,200-A)
1140 FOR DELAY=1 TO 5 :: NEXT DELAY
1150 NEXT A
1160 GOTO 1090
1170 CALL SOUND(900,-7,0)
1180 FOR D=2 TO 16
1190 CALL SCREEN(D):: FOR DEL=1 TO 5 ::
NEXT DEL
1200 NEXT D
1210 CALL DELSPRITE(ALL):: CALL CLEAR
1220 BREAK
```

Initiation au BASIC

Roger Didi

Le moyen le plus agréable d'apprendre en amateur le TI-Basic (ou Basic simple) est certainement de programmer soi-même. Aussi présentons-nous ici un programme évolutif que l'on peut étudier simplement en suivant la progression proposée, ou que l'on peut modifier à volonté en l'orientant vers des variantes qui dépendront de l'humeur du moment.

Nous avons d'ailleurs suivi une démarche similaire et, partant à l'aventure, avons abouti à un jeu amusant qui ne manque pas d'intérêt.

Au commencement, nous avons voulu créer un bonhomme... Il nous a paru intéressant de le faire marcher, puis de lui donner des couleurs. Plus tard, nous avons souhaité entendre ses pas sur la route. Quelques embûches ?... Un trou pourrait apparaître devant lui, à un endroit déterminé au hasard, et il y tomberait. On pourrait commander par la pression d'une touche l'apparition aléatoire de plusieurs trous qu'il serait possible de changer de place...

D'où le jeu qui consisterait à éviter que le bonhomme ne tombe dans un de ces trous.

Pour agrémenter le tout, quelques oranges lui donnent envie de sauter.

Nous vous laissons le soin d'imaginer une suite : par exemple un compteur permettrait de comparer les réflexes de plusieurs joueurs, une musique soulignerait une chute ou un record battu, etc... Laissez votre imagination vous conduire et canalisez-la à l'aide du Basic et de votre TI-99.

Programme 1 : construction d'un dessin

L'instruction REM (remarque) nous permet d'écrire des remarques destinées au lecteur du programme. Cette instruction et le texte qu'elle contient ne sont pas pris en compte par la machine, sauf lorsqu'elle fait le décompte de la place mémoire restant disponible ! Il est donc inutile de recopier sur votre appareil les lignes commençant par REM.

Les REMs sont utiles lors de la réalisation du programme : elles permettent de noter certains repères, de rappeler le rôle des variables ou des sous-programmes utilisés. Elles sont surtout utiles lorsque le programme est réalisé en plusieurs séances. Une fois le programme terminé, on peut supprimer toutes les REMs avant de l'enregistrer définitivement, surtout si l'on est "serré" au point de vue mémoire.

Nous présentons, en même temps que le programme, un organigramme permettant d'observer l'ensemble du déroulement du programme. Le premier rectangle contient en général les instructions initialisant le système : ici, construction du caractère numéro 100 et effacement de l'écran. Les rectangles suivants contiennent le corps même du programme. Enfin, les traits fléchés indiquent la succession des séquences du programme.

La ligne 200 du programme peut paraître insolite. Son objet est de construire la forme adoptée pour le caractère 100. Il faut se représenter ce caractère comme construit dans une grille de 8 carreaux sur 8. En traçant une ligne verticale au milieu de cette grille, on délimite des groupes de quatre caractères binaires, chaque groupe représentant une valeur de 0 à 15 ; nous associons à cette valeur le code hexadécimal correspondant de 0 à F. Les valeurs données entre guillemets dans la ligne 200 du programme 1 sont les codes hexadécimaux de chacun des 16 caractères binaires. Le caractère 100 se représente donc par :

3 0	00110000
3 6	00110110
1 9	00011001
6 D	01101101
1 4	00010100
1 2	00010010
0 9	00001001
0 0	00000000

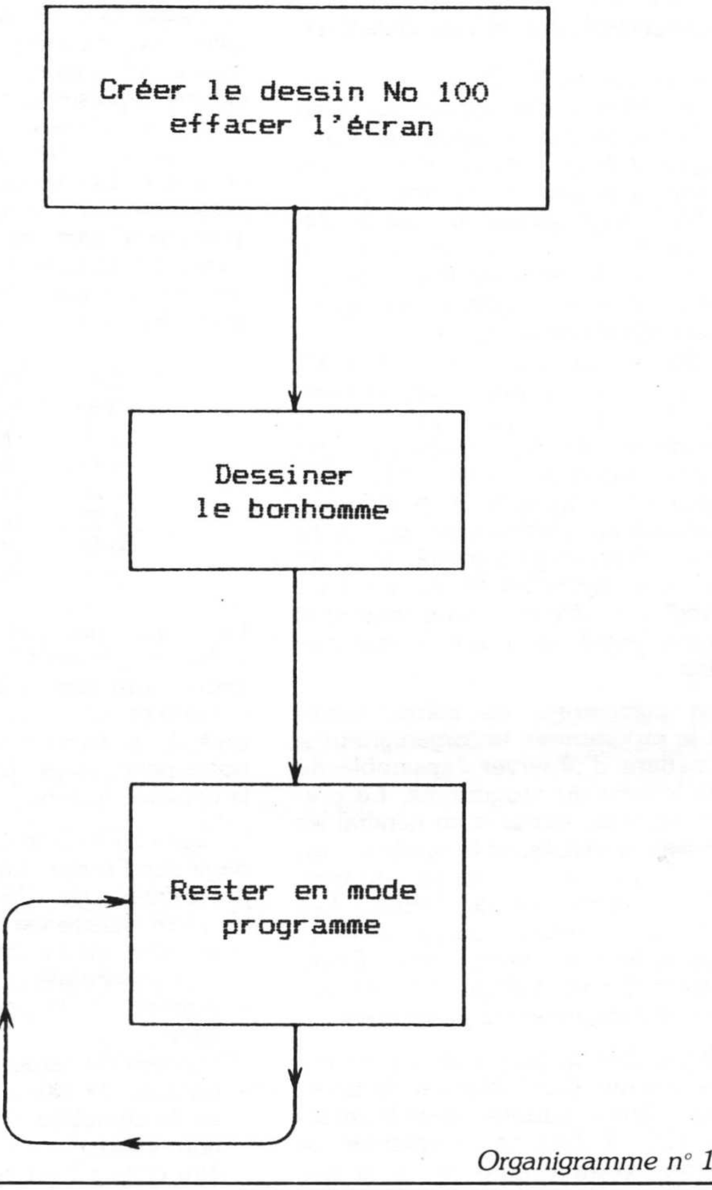
Les deux derniers caractères étant nuls, ils peuvent être omis. Nous avons ainsi rempli la grille, les cases contenant un chiffre 1 seront colorées à la couleur déterminée pour notre personnage, les autres seront à la couleur du fond.

La ligne 210 place ensuite ce personnage sur l'écran. La signification des paramètres de l'instruction CALL HCHAR utilisée est la suivante :

- numéro de la ligne dans laquelle sera représenté le caractère
- numéro de la colonne dans cette ligne
- numéro du caractère à représenter
- nombre de fois qu'il faut représenter le caractère à partir de la position indiquée (ce paramètre peut être omis s'il est égal à 1)

```
10 REM PROGRAMME NO 1
20 REM
50 REM ======================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET ROGER DIDI
80 REM ======================
90 REM
100 REM ******************
110 REM *CONSTRUCTION D'UN*
120 REM *     DESSIN     *
130 REM ******************
140 REM  On utilise CALL CHAR pour creer
le dessin, ensuite on l'imprime sur
l'ecran a l'aide  de CALL HCHAR.
150 REM
160 REM
170 REM
180 REM
190 CALL CLEAR
200 CALL CHAR(100,"3036196C141209")
210 CALL HCHAR(10,20,100,1)
220 GOTO 220
230 REM
240 REM
250 REM
260 REM  On donne le numero 100 a notre
dessin. Or 100 est deja le code d'une
lettre.
```

```
270 REM Tant que le programme tourne le
code 100 est affecte au dessin cree avec
CALL CHAR.
280 REM
290 REM Des que le programme s'arrete,
le code 100 redevient le code de la
lettre(ICI 'd')
300 REM
310 REM
320 REM On fait donc boucler le
programme en 220.
330 REM
340 REM Pour l'arreter il faut presser :
'FCTN 4'
```



Organigramme n° 1

Programme 2 : déplacement du dessin

L'écran du TI-99 contient 24 lignes, numérotées de haut en bas de 1 à 24. Chacune de ces lignes comporte 32 positions numérotées de 1 à 32 de la gauche vers la droite.

Ce programme nous permet de faire deux observations essentielles :

1 - L'utilisation des variables numériques : le bonhomme se déplace sur la ligne numéro 10. L'emplacement du caractère qui le représente sur cette ligne est donc une variable. Nous choisissons une case de la mémoire de l'ordinateur que nous représentons par la lettre X. Nous affectons ensuite à X la valeur 32, numéro du dernier caractère à droite de l'écran. Puis nous décrémentons le contenu de la case X (instruction 290 : X=X−1). Dans l'instruction CALL HCHAR, la lettre X remplace le numéro de la colonne : la machine cherche le contenu de la case X chaque fois que cette lettre est mentionnée dans le programme comme une valeur. Ainsi, à chaque tour, le bonhomme se dessine à la gauche du précédent.

2 - L'utilisation d'une boucle conditionnelle : tant que la valeur de X est supérieure à 0 (zéro), l'ordinateur continue à dessiner (ligne 270) puis à effacer (ligne 280) le dessin pour le recréer ensuite. Aussitôt que le contenu de X devient inférieur à 1, le programme s'arrête.

Observez l'organigramme : il est typique de ce genre de raisonnement. Nous le retrouverons dans presque tous les programmes que nous réaliserons. Le test est représenté par le contenu d'un losange. Pour que l'organigramme représente le mieux possible le déroulement du programme, il est préférable de marquer la sortie "OUI" horizontalement (comme un renvoi), et la sortie "NON" verticalement, ce qui indique que le programme se poursuit en cas de réponse négative. Les rectangles et les losanges se suivent donc sur une même colonne ; seuls les traits fléchés précisent l'ordre des séquences.

Programme 3 : mouvement

Ce programme résout deux problèmes classiques que l'on rencontre dans tout programme : créer une temporisation et réaliser une boucle conditionnelle.

1 - La PAUSE (lignes 290 et 300) consiste à interrompre le déroulement du programme pendant un temps donné, par exemple pour permettre à l'utilisateur de lire un message qui vient d'être affiché à l'écran. On choisit alors une case mémoire, nommée ici T (on aurait pu l'appeler DELAI pour plus de clarté) et on fait varier le contenu de cette case sans que la machine n'effectue aucun travail, avec une boucle FOR...NEXT.

2 - La boucle conditionnelle alternative : un paramètre utilisé dans une boucle prend l'une ou l'autre de deux valeurs déterminées (ici, U prend la valeur 100 pendant un tour et la valeur 101 au tour suivant). La boucle dispose alors de deux départs distincts, sachant que si le premier est emprunté il faudra "sauter" le second. Cette boucle se termine par un test : U est-il égal à 100 ? Si oui, on rejoint le second départ, sinon on prend le premier départ et on saute le second.
De cette manière, le bonhomme qui se déplace sur l'écran est représenté alternativement par le caractère de code 100 et celui de code 101 : on a

```
10 REM PROGRAMME NO 2
20 REM
50 REM =======================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET ROGER DIDI
80 REM =======================
90 REM
100 REM ******************
110 REM *DEPLACEMENT D'UN *
120 REM *      DESSIN      *
130 REM ******************
140 REM PRINCIPE :
150 REM  On imprime le dessin a un
endroit de l'ecran, puis on le remplace
par un espace, puis on le replace a un
autre endroit ETC..
160 REM
170 REM
180 REM
190 REM Le dessin evolue sur la ligne 10
de la droite vers la  gauche.
La premiere colonne est la colonne 32:
ligne 240.
200 REM
210 REM
220 REM
230 REM
240 X=32
250 CALL CLEAR
260 CALL CHAR(100,"3036196C141209")
270 CALL HCHAR(10,X,100,1)
280 CALL HCHAR(10,X,32)
290 X=X-1
300 IF X<1 THEN 320
310 GOTO 270
320 END
```

```
10 REM PROGRAMME NO 3
50 REM =======================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET ROGER DIDI
80 REM =======================
90 REM
100 REM ******************
110 REM *    MOUVEMENTS    *
120 REM ******************
130 REM PRINCIPE :
140 REM on voudrait remuer les jambes du
 coureur !
150 REM
160 REM
170 REM il suffit pour cela de creer
deux dessins du coureur. Le premier aura
les jambes dans une position et le
second dans  une autre.
180 REM Pour creer l'illusion du
mouvement on imprime tour a tour les
deux dessins.
190 REM
200 REM
210 X=31
220 CALL CLEAR
230 CALL CHAR(100,"3036196C141209")
240 CALL CHAR(101,"3036196C142340")
250 U=100
260 GOTO 280
270 U=101
280 CALL HCHAR(10,X,U,1)
290 FOR T=1 TO 50
300 NEXT T
310 CALL HCHAR(10,X,32,1)
320 X=X-1
330 IF X<1 THEN 390
340 IF U=100 THEN 270
350 GOTO 250
360 REM
370 REM
380 REM Pour que le mouvement des jambes
soit visible, on a mis une boucle
d'attente en 290-300
390 END
```

l'impression qu'il marche au lieu de se déplacer par une sorte de glissade. Une autre solution, astucieuse mais plus difficile à comprendre, consiste à écrire (sans REMs) le programme 3 bis ci-après. Cette solution utilise les variables booléennes (FAUX=0, VRAI=−1).

Programmes 4 et 5 : couleurs

Les programmes numéros 4 et 5 modifient les couleurs des caractères en fonction de la couleur de l'écran. Pour les comprendre, il suffit de se reporter aux explications concernant CALL SCREEN et CALL COLOR dans le manuel d'utilisation du TI-99.

Programme 6 : mouvement du bonhomme

Il s'agit de permettre à l'utilisateur de mettre en marche ou d'arrêter à volonté le bonhomme en appuyant sur l'une des touches "A" ou "M". Nous choisissons pour cela une case mémoire qui jouera le rôle d'un DRAPEAU (dans un aiguillage, un drapeau permet de signaler au conducteur du train la direction qu'il doit prendre !).

Nous placerons donc dans la case M de la mémoire la valeur 0 ou 1 suivant que le drapeau doit être levé (marche) ou baissé (arrêt). Il suffit ensuite de tester à chaque tour la valeur de M pour savoir si l'on doit ou non "sauter" l'instruction de variation de forme et de position du bonhomme. Nous n'avons pas écrit de remarque dans le programme avant l'instruction CALL KEY car elle demande un commentaire plus développé (voir à ce sujet l'article de Bernard Braun).

CALL KEY est une instruction qui permet de scruter le clavier pendant le déroulement du programme : deux cases mémoires, que nous avons nommées K et S, sont réservées à cette surveillance. Dans la case S, la machine inscrit la valeur 0 si aucune touche n'est pressée et la valeur 1 dans le cas contraire. En outre, S

prend la valeur −1 si la même touche reste enfoncée. La variable K, quant à elle, prend la valeur du code ASCII de la touche pressée.

Il ne reste plus qu'à gérer les conséquences de ces actions et à écrire les tests correspondants aux faits que l'on désire voir se dérouler suivant les valeurs de ces paramètres (lignes 310 et 320).

La fonction ASC permet de déterminer le code ASCII d'un caractère donné. Ainsi K=ASC("M") revient-il à écrire K=77.

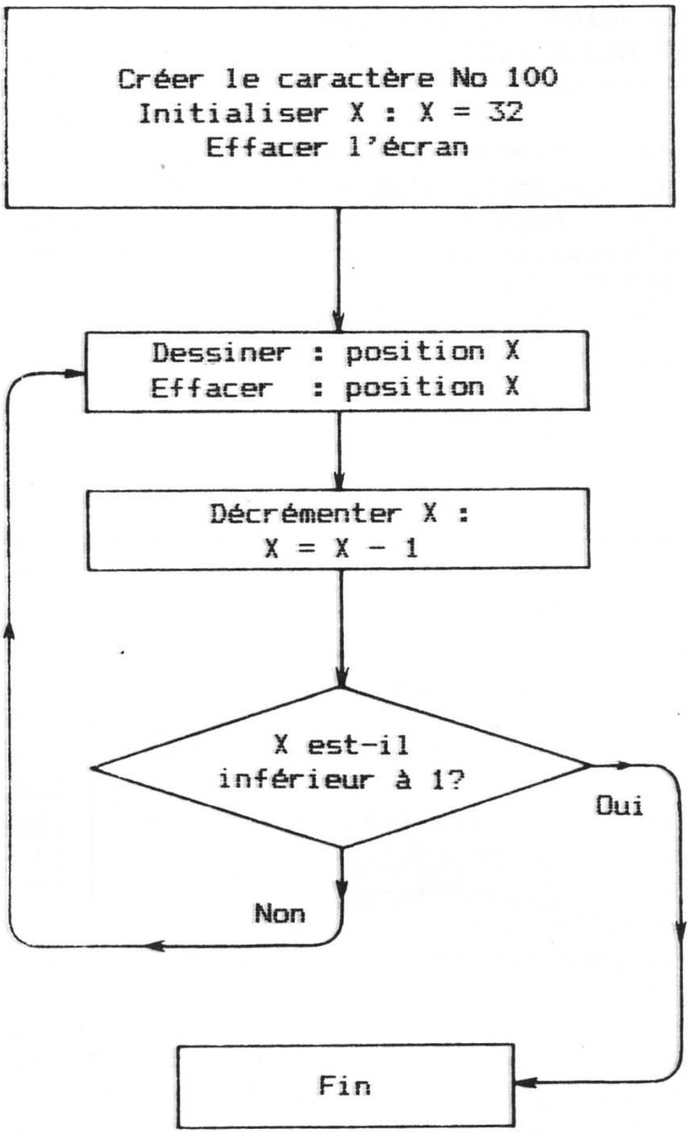
Programme 7 : bruitage et vitesse

Nous avons ici simplement complété l'utilisation de CALL KEY (lignes 230 et 250 du programme 6) pour agir sur la vitesse du bonhomme, et nous avons ajouté des bruits (ligne 390 du programme 7). Etudiez ces programmes qui n'apportent par ailleurs rien de nouveau dans notre apprentissage du Basic. Essayez cependant de construire l'organigramme du programme 7.

Programme 8 : on rencontre un trou

Nous utilisons ici les instructions RND et GOSUB, ainsi que la fonction INT.

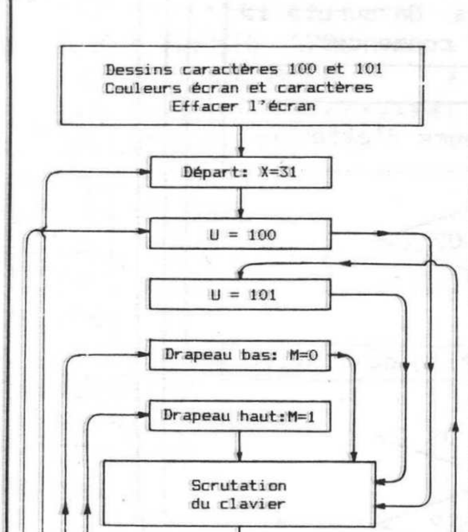
RND est une instruction qui génère un nombre aléatoire compris entre 0 et 1 (0 inclus, 1 exclus). En multipliant RND par 30, nous obtenons donc un nombre réel compris entre 0 et 30 (exclus). Mais ce nombre n'est pas entier. La fonction INT (de l'anglais INTEGER = partie entière) donne le plus grand nombre entier inférieur à l'argument entre parenthèses. Ainsi INT(3.8)=3 ; par contre, INT(−3.8)=−4.



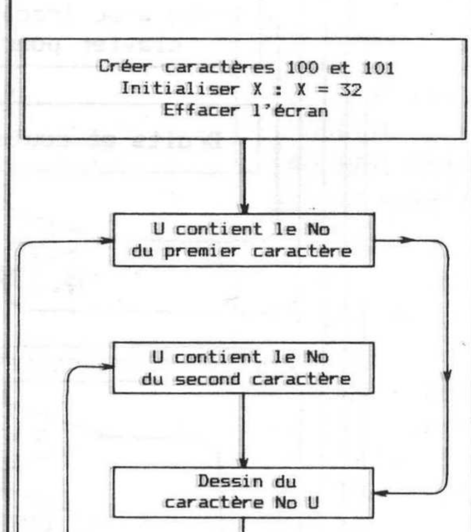
Organigramme n° 2

```
10 REM PROGRAMME NO 4
50 REM ======================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET ROGER DIDI
80 REM ======================
90 REM
100 REM *******************
110 REM *DESSIN ET COULEURS*
120 REM *******************
130 REM PRINCIPE
140 REM Le coureur est vert sur fond
blanc .
150 REM
160 REM
170 REM On utilise l'instruction 'CALL
COLOR' : les numeros 100  et 101 sont
dans la serie 9.
180 REM
190 CALL COLOR(9,13,15)
200 CALL CLEAR
210 REM
220 CALL CHAR(100,"3036196C141209")
230 CALL CHAR(101,"3036196C142340")
240 X=31
250 U=100
260 GOTO 280
270 U=101
280 CALL HCHAR(10,X+1,32,1)
290 CALL HCHAR(10,X,U,1)
300 X=X-1
310 IF X<1 THEN 360
320 FOR T=1 TO 50
330 NEXT T
340 IF U=100 THEN 270
350 GOTO 250
360 END
```

```
10 REM PROGRAMME NO 5
50 REM =====================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET ROGER DIDI
80 REM =====================
90 REM
100 REM ******************
110 REM *DEPLACEMENT D'UN *
115 REM *     DESSIN      *
120 REM ******************
130 REM
140 REM
150 REM Il vaudrait mieux que le dessin
ait la couleur de l'ecran
160 REM Modifions la couleur de l'ecran
(CALL SCREEN)
170 CALL COLOR(9,13,15)
180 CALL SCREEN(15)
190 CALL CLEAR
200 CALL CHAR(100,"3036196C141209")
210 CALL CHAR(101,"3036196C142340")
220 X=31
230 U=100
240 GOTO 260
250 U=101
260 CALL HCHAR(10,X+1,32,1)
270 CALL HCHAR(10,X,U,1)
280 X=X-1
290 IF X<1 THEN 340
300 FOR T=1 TO 50
310 NEXT T
320 IF U=100 THEN 250
330 GOTO 230
340 END
```



Organigramme n° 3



Organigramme n° 6

Pour obtenir un nombre entier compris entre a et b il suffit d'écrire INT(RND⋆(b−a)+a). La ligne 260 du programme 7 générera donc un nombre entier compris entre 1 et 30. L'utilisation de GOSUB est ici un peu forcée. On pourrait en faire l'économie. Mais nous profitons de ce programme pour en montrer l'usage : le sous-programme dessinant le trou est utilisé deux fois par le programme principal (lignes 280 et 780 du programme 8). Le sous-programme commençant à la ligne 730 et se terminant à la ligne 760 par RETURN permet de n'écrire qu'une fois l'enchaînement d'instructions qu'il contient. Ainsi GOSUB est-il un renvoi avec esprit de retour : le contrôle est envoyé à la première ligne du sous-programme, mais, aussitôt le RETURN rencontré, il reviendra à la première instruction qui suit le GOSUB.

Programme 9 : quatre trous disposés au hasard

Quatre trous, dont les positions sont dans les cases mémoire C1, C2, C3 et C4, sont placés sur chacun des quarts de la route :

C1=INT(RND⋆8)+1
position aléatoire entre 1 et 8
C=INT(RND⋆8)+9
position aléatoire entre 9 et 16
C3=INT(RND⋆8)+17
position aléatoire entre 17 et 24
C4=INT(RND⋆8)+25
position aléatoire entre 25 et 32

Un sous-programme (lignes 550 à 590 du programme 9) détermine la position de ces quatre trous. Un autre sous-programme (lignes 600 à 660) dessine alors la route et les quatre trous.

La ligne 360 demande un commentaire détaillé. Commençons d'abord

par comprendre comment la machine exprime qu'une expression est vraie ou fausse. Pour cela frappez les instructions suivantes et observez la réponse :

PRINT 2=2	Rép : −1
PRINT 2=3	Rép : 0
PRINT 5<4	Rép : 0
PRINT 5>4	Rép : −1
PRINT (2=3)*6	Rép : 0
PRINT (5>4)*(8=8)*7	Rép : 7

Le résultat d'une comparaison est donc −1 lorsque cette comparaison est VRAIE et 0 dans le cas contraire. Il est intéressant de noter que ces résultats peuvent être utilisés dans une opération comportant d'autres signes opératoires.

Frappez maintenant les lignes suivantes et observez les réponses de la machine :

PRINT (2=2)*(4<5)*(3<7)
Réponse : −1
PRINT (2−3)*(5>7)*(8>9)
Réponse : 0

Si les trois expressions sont vraies, la réponse est non nulle; mais si l'une au moins d'entre elles est fausse, alors la réponse est 0 ! Essayez de bien comprendre ces particularités avant de lire la suite. Au besoin, faites d'autres essais...

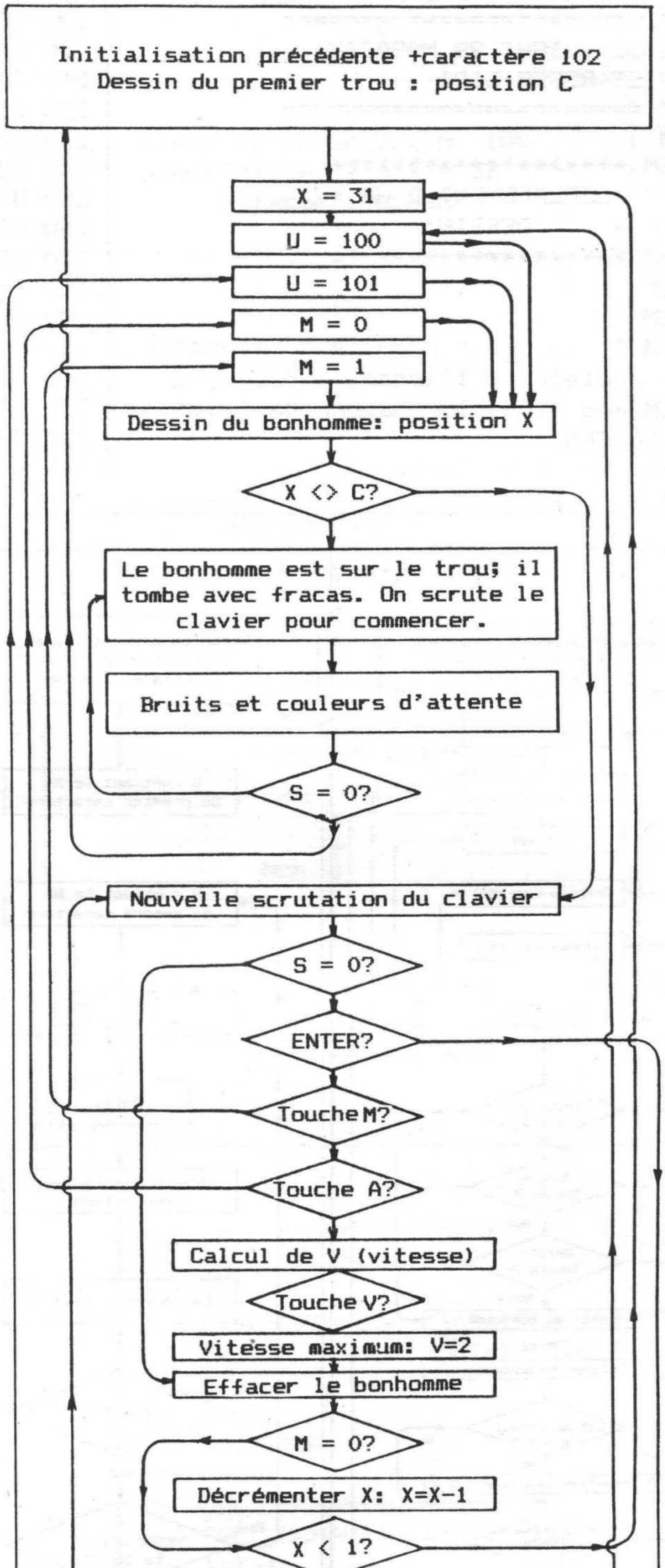
Vous pouvez alors constater que le signe "*" dans une instruction IF constitue le ET logique : la condition est vraie si l'une ET l'autre des expressions sont vraies. En revanche, si vous utilisez le signe "+", il constitue le OU logique. Ainsi, si vous écrivez :
IF (A=B)+(A=C)
la condition sera vraie si l'une OU l'autre des expressions qui la compose est vraie.

Etudions maintenant le fonctionnement de l'instruction composée IF...THEN. La machine teste l'expression comprise entre IF et THEN :

- si cette expression est nulle, elle la considère comme fausse et passe à la ligne suivante du programme, sans exécuter ce qui se trouve derrière THEN ;
- si cette expression est différente de zéro, elle la considère comme vraie et exécute ce qui est derrière THEN.

Cette précision doit maintenant vous permettre de mieux comprendre la ligne 360 du programme 9. En effet, si l'expression comprise entre IF et THEN est nulle, c'est que l'une des valeurs C1, C2, C3 ou C4 est égale à X ; il y a donc coïncidence entre la position du bonhomme et la position de l'un des trous ! D'où la conséquence...

Le reste du programme est simple à comprendre. Nous vous laissons le soin de réaliser l'organigramme complet de ce programme, comme d'ailleurs des suivants : cela peut être un excellent exercice...



Organigramme n° 8

Programme 10 : quatre trous et des oranges

Nous voyons apparaître ici pour la première fois une variable indicée : B(1), B(2), B(3) et B(4) sont les positions des quatre oranges (du moins la colonne dans laquelle elles se trouvent). Des variables indicées peuvent notamment être utilisées dans des boucles FOR...NEXT, comme dans ce sous-programme (lignes 870 à 910) qui dessine les oranges. Remarquons que l'altitude des oranges est aussi aléatoire et comprise entre 4 et 8. Etudiez ce programme et construisez son organigramme...

Programme 11 : sauter pour une orange

Lorsque le bonhomme se trouve au-dessous d'une orange, on appuie sur la barre d'espacement (code ASCII 32), ce qui le fait sauter et fait disparaître la ou les oranges qui l'intéresse. Observez pour cela le test qui se trouve à la ligne 1000.

Le reste du programme comporte peu de difficultés et il vous est possible de le comprendre entièrement. Alors bon courage...

Ecrivez-nous pour nous dire si cette méthode d'approche de l'informatique vous convient, et proposez-nous des idées pour l'améliorer. D'avance merci !

```
10 REM PROGRAMME NO 6
20 REM
50 REM ========================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET ROGER DIDI
80 REM ========================
90 REM
100 REM ******************
110 REM * MARCHE-ARRET DU *
115 REM *   BONHOMME      *
120 REM ******************
130 CALL COLOR(9,13,15)
140 CALL SCREEN(15)
150 CALL CLEAR
160 CALL CHAR(100,"3036196C141209")
170 CALL CHAR(101,"3036196C142340")
180 X=31
190 U=100
200 GOTO 290
210 U=101
220 GOTO 290
230 M=0
240 GOTO 290
250 M=1
260 REM
270 REM
280 REM La touche M fait marcher le
bonhomme. La touche A l'arrete.
290 CALL KEY(0,K,S)
300 IF S=0 THEN 340
310 IF K=ASC("M")THEN 250
320 IF K=ASC("A")THEN 230
330 REM
340 CALL HCHAR(10,X,U,1)
350 FOR DELAI=1 TO 30
360 NEXT DELAI
370 CALL HCHAR(10,X,32,1)
380 IF M=0 THEN 400
390 X=X-1
400 IF X<1 THEN 180
410 IF U=100 THEN 210
420 GOTO 190
430 REM 'M' joue le role d'un drapeau.
Quand M=1, il est haut. Quand M=0, il
est bas.
440 REM ESSAYEZ D'ARRETER LES PATTES DU
BONHOMME!!!
```

```
10 REM PROGRAMME NO 7
50 REM ========================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET ROGER DIDI
80 REM ========================
90 REM
100 REM ******************
110 REM *  BRUITAGE  ET  *
115 REM *    VITESSE     *
120 REM ******************
130 REM
140 REM PRINCIPE : Un bruit de pas
accompagne la course.
150 REM la vitesse VARIE sous l'action
d'une touche quelconque. Mais V accelere
le mouvement!!
160 REM
170 CALL CHAR(94,"0")
180 REM Le caractere 94 est un  blanc
(pour la route).
190 CALL COLOR(8,12,12)
200 CALL COLOR(9,15,13)
210 CALL SCREEN(13)
220 CALL CLEAR
230 REM Trois lignes de blancs font une
route !!!
240 CALL HCHAR(11,1,94,96)
250 CALL CHAR(100,"3036196C141209")
260 CALL CHAR(101,"3036196C142340")
270 X=31
280 U=100
290 GOTO 350
300 U=101
310 GOTO 350
320 M=0
330 GOTO 350
340 M=1
350 CALL HCHAR(10,X,U,1)
360 FOR DELAI=1 TO V
370 REM Plus V est grand plus la vitesse
est faible.
380 NEXT DELAI
390 CALL SOUND(5,-4,9,110,7)
400 CALL KEY(0,K,S)
410 IF S=0 THEN 490
420 IF K=ASC("M")THEN 340
```

```
430 IF K=ASC("A")THEN 320
440 REM La vitesse V se calcule a partir
du code de la touche pressee.
450 V=(K-32)*10
460 IF K<>ASC("V")THEN 490
470 REM Si c'est "V" la vitesse est
MAXIMUM : V=2.
480 V=2
490 CALL HCHAR(10,X,32,1)
500 IF M=0 THEN 520
510 X=X-1
520 IF X<1 THEN 270
530 IF U=100 THEN 300
540 GOTO 280
```

```
10 REM PROGRAMME NO 8
50 REM ======================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET ROGER DIDI
80 REM ======================
90 REM
100 REM ******************
110 REM *  ON  RENCONTRE  *
115 REM *    UN TROU      *
120 REM ******************
130 REM
140 REM PRINCIPE: Le bonhomme est arrete
par un trou creuse dans la chaussee
aleatoirement...
150 REM Il tombe dans le trou et une
musique l'accompagne!!
160 REM
170 CALL CHAR(94,"0")
180 CALL CHAR(100,"3036196C141209")
190 CALL CHAR(102,"0912146C193630")
200 CALL CHAR(101,"3036196C142340")
210 CALL COLOR(8,12,12)
220 CALL COLOR(9,15,13)
230 CALL SCREEN(13)
240 CALL CLEAR
250 REM La lettre C contient la position
aleatoire du trou.
260 C=INT(RND*30)+1
270 REM Le dessin du trou est fait dans
un  sous-programme.
280 GOSUB 730
290 X=31
300 U=100
310 GOTO 370
320 U=101
330 GOTO 370
340 M=0
350 GOTO 370
360 M=1
370 CALL HCHAR(10,X,U)
380 IF X<>C THEN 540
390 REM Si X=C le bonhomme est sur le
trou, il doit donc tomber.
400 CALL SCREEN(2)
410 FOR L=10 TO 12
420 CALL HCHAR(L,X,102,1)
430 CALL HCHAR(L,X,32,1)
440 NEXT L
450 CALL HCHAR(13,X,102,1)
460 REM Un peu du bruit!!
470 CALL SOUND(100,-7,0,220,3)
480 CALL KEY(0,K,S)
490 REM Du fond, il voit varier la
couleur de l'ecran .
500 CALL SCREEN(INT(RND*15)+1)
510 CALL SOUND(10,INT(RND*200)+110,3,-1,
2)
520 IF S=0 THEN 480
530 GOTO 230
540 FOR DELAI=1 TO V
550 NEXT DELAI
560 CALL SOUND(5,-4,9,110,7)
570 CALL KEY(0,K,S)
580 IF S=0 THEN 660
590 REM La touche ENTER (code 13)
provoque  le deplacement du trou.
600 IF K=13 THEN 770
610 IF K=ASC("M")THEN 360
620 IF K=ASC("A")THEN 340
630 V=(K-32)*10
640 IF K<>ASC("V")THEN 660
650 V=2
660 CALL HCHAR(10,X,32)
670 IF M=0 THEN 690
680 X=X-1
690 IF X<1 THEN 290
700 IF U=100 THEN 320
710 GOTO 300
720 REM Sous-programme pour le dessin du
trou :
730 CALL CLEAR
740 CALL HCHAR(11,1,94,96)
750 CALL VCHAR(11,C,32,3)
760 RETURN
770 C=INT(RND*30)+1
780 GOSUB 730
790 GOTO 570
```

```
10 REM PROGRAMME NO 9
20 REM
50 REM ======================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET ROGER DIDI
80 REM ======================
90 REM
100 GOSUB 550
110 REM ******************
120 REM *   QUATRE TROUS  *
125 REM *    AU HASARD    *
130 REM ******************
```

```
140 REM
150 REM PRINCIPE : Quatre trous sont
creuses aleatoirement, il faut les
eviter.
160 REM La touche ENTER dont le code est
13, change leur place.
170 REM
180 CALL CHAR(94,"0")
190 CALL COLOR(8,12,12)
200 CALL COLOR(9,15,13)
210 CALL SCREEN(13)
220 CALL CLEAR
230 GOSUB 600
240 CALL CHAR(100,"3036196C141209")
250 CALL CHAR(102,"0912146C193630")
260 CALL CHAR(101,"3036196C142340")
270 X=31
280 U=100
290 GOTO 350
300 U=101
310 GOTO 350
320 M=0
330 GOTO 350
340 M=1
350 CALL HCHAR(10,X,U,1)
360 IF (X<>C1)*(X<>C2)*(X<>C3)*(X<>C4)TH
EN 380
370 GOTO 700
380 FOR DELAI=1 TO V
390 NEXT DELAI
400 CALL SOUND(5,-4,9,110,7)
410 CALL KEY(0,K,S)
420 IF S=0 THEN 490
430 IF K=13 THEN 670
440 IF K=ASC("M")THEN 340
450 IF K=ASC("A")THEN 320
460 V=(K-32)*10
470 IF K<>ASC("V")THEN 490
480 V=2
490 CALL HCHAR(10,X,32,1)
500 IF M=0 THEN 520
510 X=X-1
520 IF X<1 THEN 270
530 IF U=100 THEN 300
540 GOTO 280
550 C1=INT(RND*8)+1
560 C2=INT(RND*8)+9
570 C3=INT(RND*8)+17
580 C4=INT(RND*8)+25
590 RETURN
600 CALL CLEAR
610 CALL HCHAR(11,1,94,96)
620 CALL VCHAR(11,C1,32,3)
630 CALL VCHAR(11,C2,32,3)
640 CALL VCHAR(11,C3,32,3)
650 CALL VCHAR(11,C4,32,3)
660 RETURN
670 GOSUB 550
680 GOSUB 600
690 GOTO 410
700 CALL SCREEN(2)
710 CALL HCHAR(10,X,102,1)
720 CALL HCHAR(10,X,32,1)
730 CALL HCHAR(11,X,102,1)
740 CALL HCHAR(11,X,32,1)
750 CALL HCHAR(12,X,102,1)
760 CALL HCHAR(12,X,32,1)
770 CALL HCHAR(13,X,102,1)
780 CALL SOUND(100,-7,0,220,3)
790 CALL KEY(0,K,S)
800 CALL SCREEN(INT(RND*15)+1)
810 CALL SOUND(10,INT(RND*200)+110,3,-1,
2)
820 IF S=0 THEN 790
830 GOTO 100
```

```
10 REM  PROGRAMME NO 10
20 REM
30 REM ======================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET ROGER DIDI
80 REM ======================
90 REM
100 GOSUB 600
110 REM ******************
120 REM * QUATRE TROUS ET *
125 REM *  DES ORANGES    *
130 REM ******************
140 REM
150 REM PRINCIPE : Quatre oranges
apparaissent aleatoirement.
160 REM Elles changent de place avec les
trous.
170 CALL CHAR(128,"0")
180 CALL COLOR(13,13,13)
190 CALL CHAR(94,"0")
200 CALL COLOR(8,12,12)
210 CALL COLOR(9,15,6)
220 CALL SCREEN(6)
230 CALL CLEAR
240 CALL HCHAR(11,1,128,448)
250 GOSUB 870
260 GOSUB 650
270 CALL CHAR(100,"3036196C141209")
280 CALL CHAR(102,"0912146C193630")
290 CALL CHAR(101,"3036196C142340")
300 CALL CHAR(123,"3C5EDFEFFFFF7E1C")
310 CALL COLOR(12,10,6)
320 X=31
330 U=100
340 GOTO 400
350 U=101
360 GOTO 400
370 M=0
380 GOTO 400
390 M=1
```

```
400 CALL HCHAR(10,X,U,1)
410 IF (X<>C1)*(X<>C2)*(X<>C3)*(X<>C4)TH
EN 430
420 GOTO 750
430 FOR DELAI=1 TO V
440 NEXT DELAI
450 CALL SOUND(5,-4,9,110,7)
460 CALL KEY(0,K,S)
470 IF S=0 THEN 540
480 IF K=13 THEN 720
490 IF K=ASC("M")THEN 390
500 IF K=ASC("A")THEN 370
510 V=(K-32)*10
520 IF K<>ASC("V")THEN 540
530 V=2
540 CALL HCHAR(10,X,32,1)
550 IF M=0 THEN 570
560 X=X-1
570 IF X<1 THEN 320
580 IF U=100 THEN 350
590 GOTO 330
600 C1=INT(RND*8)+1
610 C2=INT(RND*8)+9
620 C3=INT(RND*8)+17
630 C4=INT(RND*8)+25
640 RETURN
650 REM
660 CALL HCHAR(11,1,94,96)
670 CALL VCHAR(11,C1,32,3)
680 CALL VCHAR(11,C2,32,3)
690 CALL VCHAR(11,C3,32,3)
700 CALL VCHAR(11,C4,32,3)
710 RETURN
720 GOSUB 600
730 GOSUB 650
740 GOTO 460
750 CALL SCREEN(2)
760 FOR L=10 TO 12
770 CALL HCHAR(L,X,102,1)
780 CALL HCHAR(L,X,32,1)
790 NEXT L
800 CALL HCHAR(13,X,102,1)
810 CALL SOUND(100,-7,0,220,3)
820 CALL KEY(0,K,S)
830 CALL SCREEN(INT(RND*15)+1)
840 CALL SOUND(10,INT(RND*200)+110,3,-1,
2)
850 IF S=0 THEN 820
860 GOTO 100
870 FOR I=1 TO 4
880 B(I)=INT(RND*32)+1
890 CALL HCHAR(INT(RND*4)+4,B(I),123)
900 NEXT I
910 RETURN
```

```
10 REM  PROGRAMME NO 11
50 REM ======================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET ROGER DIDI
80 REM ======================
90 REM
100 GOSUB 600
110 REM ******************
120 REM * SAUTER POUR UNE *
125 REM *     ORANGE      *
130 REM ******************
140 REM
150 REM PRINCIPE : En appuyant sur la
barre on fait sauter le petit bonhomme.
160 REM Il peut alors prendre une
orange!!
170 CALL CHAR(128,"0")
180 CALL COLOR(13,13,13)
190 CALL CHAR(94,"0")
200 CALL COLOR(8,12,12)
210 CALL COLOR(9,15,6)
220 CALL SCREEN(6)
230 CALL CLEAR
235 GOSUB 890
240 CALL HCHAR(11,1,128,448)
250 GOSUB 650
260 CALL CHAR(100,"3036196C141209")
270 CALL CHAR(102,"0912146C193630")
280 CALL CHAR(101,"3036196C142340")
290 CALL CHAR(123,"3C5EDFEFFFFF7E1C")
300 CALL COLOR(12,10,6)
310 X=31
320 U=100
330 GOTO 390
340 U=101
350 GOTO 390
360 M=0
370 GOTO 390
380 M=1
390 CALL HCHAR(10,X,U,1)
400 IF (X<>C1)*(X<>C2)*(X<>C3)*(X<>C4)TH
EN 420
410 GOTO 750
420 FOR DELAI=1 TO V
430 NEXT DELAI
440 CALL SOUND(5,-4,9,110,7)
450 CALL KEY(0,K,S)
460 IF S=0 THEN 540
470 IF K=13 THEN 720
480 IF K=32 THEN 940
490 IF K=ASC("M")THEN 380
500 IF K=ASC("A")THEN 360
510 V=(K-32)*10
520 IF K<>ASC("V")THEN 540
530 V=2
540 CALL HCHAR(10,X,32,1)
550 IF M=0 THEN 570
560 X=X-1
570 IF X<1 THEN 310
```

```
580 IF U=100 THEN 340
590 GOTO 320
600 C1=INT(RND*8)+1
610 C2=INT(RND*8)+9
620 C3=INT(RND*8)+17
630 C4=INT(RND*8)+25
640 RETURN
650 REM
660 CALL HCHAR(11,1,94,96)
670 CALL VCHAR(11,C1,32,3)
680 CALL VCHAR(11,C2,32,3)
690 CALL VCHAR(11,C3,32,3)
700 CALL VCHAR(11,C4,32,3)
710 RETURN
720 GOSUB 600
730 GOSUB 650
740 GOTO 450
750 CALL SCREEN(2)
770 FOR L=10 TO 12
780 CALL HCHAR(L,X,102,1)
790 CALL HCHAR(L,X,32,1)
800 NEXT L
820 CALL HCHAR(13,X,102,1)
830 CALL SOUND(100,-7,0,220,3)
840 CALL KEY(0,K,S)
850 CALL SCREEN(INT(RND*15)+1)
860 CALL SOUND(10,INT(RND*200)+110,3,-1,
2)
870 IF S=0 THEN 840
880 GOTO 100
890 FOR I=1 TO 4
900 B(I)=INT(RND*32)+1
910 CALL HCHAR(INT(RND*4)+4,B(I),123)
920 NEXT I
930 RETURN
940 CALL HCHAR(10,X,32)
950 CALL HCHAR(9,X,U)
960 CALL HCHAR(9,X,32)
970 CALL HCHAR(8,X,U)
980 CALL HCHAR(8,X,32)
990 CALL VCHAR(1,X,32,10)
1000 IF (X<>B(1))*(X<>B(2))*(X<>B(3))*(X
<>B(4))THEN 1020
1010 CALL SOUND(30,200,1)
1020 CALL HCHAR(9,X,U)
1030 CALL HCHAR(9,X,32)
1040 CALL HCHAR(10,X,U)
1050 GOTO 450
```

Le jeu du Pendu

Edouard Jouan

Voici un jeu dont le principe est très simple : l'ordinateur choisit parmi une liste contenue dans le programme un mot qu'il demande au joueur de reconstituer. Si la lettre proposée par le joueur existe dans le mot, elle s'inscrit à sa place dans le mot à chaque endroit (s'il y en a plusieurs) où elle doit apparaître. Dans le cas contraire, un élément du pendu s'ajoute. Lorsque le mot est complet, la partie est gagnée; elle est perdue si le pendu est complet avant la reconstitution totale du mot.

Présentation du jeu

La première tâche d'un programme de jeu bien conçu consiste, bien entendu, à présenter les règles. Les lignes 220 à 310 assurent ce service. Remarquez-y la présence de l'instruction DISPLAY : cette instruction n'est pas disponible en Basic simple, mais seulement en Basic étendu. Seuls donc ceux des lecteurs qui disposent de ce module pourront faire fonctionner ce programme tel quel.

Les lignes 140 à 180 utilisent l'instruction CALL KEY. Cette instruction doit toujours être suivie de trois valeurs :

- la première de ces valeurs est un indice donnant la composition du clavier. L'indice 0 laisse au clavier sa composition normale, telle qu'elle apparait sur les touches.

- la seconde valeur sera le code ASCII de la première touche frappée, par exemple 78 si vous tapez N ou 79 si vous tapez O.

- la troisième valeur enfin, indique si une touche a été frappée. Elle est égale à 0 tant que cet évènement ne s'est pas produit. Cela explique la ligne 150 : tant que la variable ETAT est nulle, on revient à la ligne 140, et ainsi de suite... De la sorte, le programme tourne sur lui-même en attendant que le joueur ait indiqué sa décision.

La ligne 170 vérifie que le caractère frappé est bien "N". Dans ce cas, le programme saute l'édition des règles et passe à la ligne 350.

La ligne 180 vérifie ensuite que ce caractère frappé est bien "O" (dans le cas où il n'était pas "N"). Si ce n'est pas le cas, le programme retourne à la ligne 140. Il est donc impossible de poursuivre l'exécution de ce programme sans frapper "N" ou "O".

Création des formes et des couleurs

Les lignes 350 à 590 composent les formes et couleurs des caractères graphiques utilisés par le programme : les caractères 96 à 98 seront utilisés pour le dessin de la potence (les numéros de caractères sont les codes ASCII de ces caractères). Les caractères 104 à 121 seront utilisés pour la composition du pendu.

Remarquez que l'instruction CALL COLOR est suivie, elle aussi, de trois valeurs :
- la première est la série de caractères concernée : reportez-vous aux

codes ASCII du TI , et vous constaterez que ces codes sont groupés par séries de huit. Lorsque vous déterminez une couleur, elle concernera l'ensemble de la série de caractères.
- la seconde est la couleur choisie pour le caractère lui-même.
- la troisième enfin indique la couleur du fond sur lequel s'inscrira le caractère.

Lecture du "lexique"

Les lignes 620 à 660 lisent les mots placés dans les lignes de 2100 à 2190. Ces dernières contiennent le "lexique" du programme, c'est-à-dire la liste des mots qu'il connait. Le fait remarquable de l'instruction DATA est qu'elle n'obéit pas à la même loi que les autres instructions du Basic : elle peut être disposée n'importe où dans le programme. Plusieurs instructions DATA disposées dans le programme ont le même effet que s'il y en avait une seule contenant toutes les données; de plus ces instructions peuvent être dispersées : seul compte l'ordre dans lequel elles se trouvent les unes par rapport aux autres.

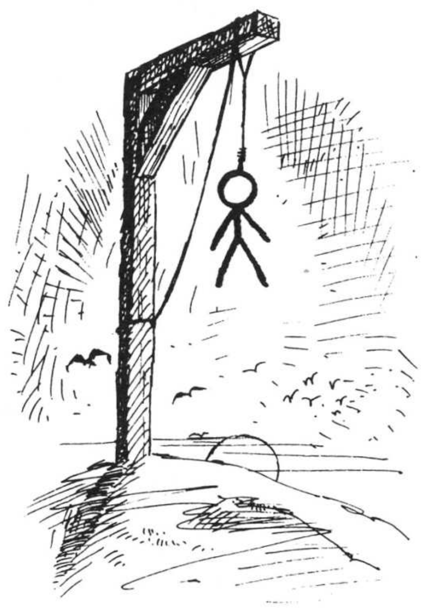
Le programme commence donc par ouvrir un tableau, appelé "MOT$", et y range les 100 mots du lexique.

Corps du programme

Après cette phase d'initialisation, nous arrivons maintenant au corps du programme. Celui-ci demande tout d'abord l'enfoncement d'une touche avant de commencer la suite de la procédure; puis il dessine la potence du pendu (lignes 940 à 970).

Le choix du mot à deviner est la conséquence de la ligne 670 et des lignes 840 à 850 : RANDOMIZE assure que le premier choix ne sera pas le même à chaque lancement du programme. La ligne 840 donne à la variable J une valeur aléatoire comprise entre 1 et 100, et la ligne 850 donne à M$ la valeur du mot de rang J dans le tableau MOT$.

La ligne 860 dispose autant de tirets



qu'il y a de lettres dans le mot à trouver. Les lignes 900 à 930 initialisent des variables et les lignes 940 à 970 demandent l'entrée d'une lettre. De 980 à 990, on dispose cette lettre à la suite de la question posée. De 1000 à 1020, on examine si la lettre proposée n'a pas déjà été donnée. Pour cela, on avait ouvert un tableau E$, de 30 éléments, dans lequel les lettres successivement proposées ont été rangées. En cas de concordance, on envoie le programme à la ligne 2020 où un message est adressé au joueur.

Le programme inscrit la lettre dans la partie droite de l'écran (1030), puis examine si elle correspond à une des lettres du mot recherché (1080 à 1100). Si c'est le cas, il va à la ligne 1180 qui inscrit la lettre à sa place dans le mot; on examine en 1210 si elle est la dernière lettre à trouver (fin gagnante du jeu), ou non (retour à la boucle 1080-1100)

Si la lettre proposée ne correspond pas à une lettre du mot, on se dirige (ligne 1170) vers la partie suivante du pendu à faire apparaître à l'écran. La procédure commençant à la ligne 1760 consiste à afficher la dernière partie du pendu (sa jambe gauche) et se termine donc par la procédure de fin perdante du jeu.

Dessin du pendu

La tête se dessine grâce aux lignes 1350 à 1420. Les épaules sont créées par les lignes 1430 à 1480. On voit ensuite apparaître le bras droit (1490 à 1540), le bras gauche, le corps, la jambe droite, et enfin la jambe gauche.

Les sons

Les sons sont générés par l'instruction CALL SOUND. Leur utilité dans ce programme est triple :

- souligner la bonne perception par l'ordinateur de l'enfoncement d'une touche demandé par une instruction CALL KEY (c'est le cas par exemple à la ligne 160). Le caractère de la touche enfoncée n'apparaît en effet pas à l'écran et, sans ce trait sonore, le manipulateur pourrait être tenté de renouveler sa manoeuvre.
- signaler au joueur l'apparition à l'écran d'un message qui lui est destiné. Les lignes 1130 et 2030 assurent ce service.
- créer une musique, simple et courte pour souligner la victoire ou la défaite du joueur (lignes 1310 à 1330 et 1840 à 1870, respectivement)

Conclusion

Le principe de ce jeu est en fait très classique pour les jeux de devinette. Vous pourrez aisément compléter le lexique du programme en ajoutant des lignes de DATA. Il convient alors de modifier en conséquence les lignes 620 et 840 pour tenir compte de l'accroissement du nombre de mots disponibles.

Le graphisme aussi ne demande qu'à être amélioré. Il est ici d'une grande simplicité et ne sollicite que votre imagination pour devenir quelque chose de magnifique. Alors, à vos claviers...

```
50 REM =====================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET TISOFT-HOME
80 REM =====================
90 REM
100 CALL CLEAR
110 CALL SCREEN(12)
120 DISPLAY AT(3,8):"JEU DU PENDU"
130 DISPLAY AT(6,1):"VOULEZ-VOUS LES REG
LES(O/N) ? "
140 CALL KEY(0,CODE,ETAT)
150 IF ETAT=0 THEN 140
160 CALL SOUND(100,2000,2)
170 IF CODE=78 THEN 350
180 IF CODE=79 THEN 220 ELSE 140
```

```
190 REM ********************
200 REM ENONCE DES REGLES
210 REM ********************
220 DISPLAY AT(8,3):"VOUS DEVEZ DEVINER
UN MOT"
230 DISPLAY AT(9,1):"DONT LES LETTRES SO
NT"
240 DISPLAY AT(10,1):"REPRESENTEES PAR D
ES TIRETS."
250 DISPLAY AT(11,3):"VOUS PROPOSEZ UNE
LETTRE."
260 DISPLAY AT(12,1):"SI ELLE FIGURE DAN
S LE MOT,"
270 DISPLAY AT(13,1):"ELLE APPARAITRA A
SA PLACE."
280 DISPLAY AT(14,1):"SINON UN ELEMENT D
U PENDU"
290 DISPLAY AT(15,1):"S'AJOUTERA AU DESS
IN."
300 DISPLAY AT(16,3):"LE JEU EST PERDU S
I"
310 DISPLAY AT(17,1):"LE PENDU EST COMPL
ET."
320 REM ********************
330 REM FORME DES CARACTERES
340 REM ********************
350 CALL CHAR(97,"0101010101010101")
360 CALL CHAR(98,"8080808080808080")
370 CALL CHAR(96,"FFFFFFFFFFFFFFFF")
380 CALL COLOR(9,14,1)
390 CALL CHAR(104,"0F10204080808080")
400 CALL CHAR(107,"F008040201010101")
410 CALL CHAR(105,"8080804020100807")
420 CALL CHAR(106,"01010102040810E0")
430 CALL CHAR(108,"010101FF01010101")
440 CALL CHAR(109,"808080FF80808080")
450 CALL CHAR(110,"0000000F10204080")
460 CALL CHAR(111,"0102040810204080")
470 CALL CHAR(112,"000000F008040201")
480 CALL CHAR(113,"8040201008040201")
490 CALL CHAR(114,"0303030303030303")
500 CALL CHAR(115,"C0C0C0C0C0C0C0C0")
510 CALL CHAR(116,"0102020404080810")
520 CALL CHAR(117,"1020204040808000")
530 CALL CHAR(118,"0000000000000001")
540 CALL CHAR(119,"8040402020101008")
550 CALL CHAR(120,"0804040202010100")
560 CALL CHAR(121,"0000000000000080")
570 CALL COLOR(10,10,1)
580 CALL COLOR(11,10,1)
590 CALL COLOR(12,10,1)
600 REM ********************
605 REM MEMORISATION DU LEXIQUE
610 REM ********************
620 DIM MOT$(100)
630 FOR I=1 TO 100
640 READ Z$
650 MOT$(I)=Z$
660 NEXT I
670 RANDOMIZE
680 IF CODE=78 THEN 760
690 DISPLAY AT(24,1):"PRESSEZ UNE TOUCHE
"
700 CALL KEY(0,A,B)
710 IF B=0 THEN 700
720 CALL SOUND(100,2000,2)
730 REM ********************
740 REM POTENCE
750 REM ********************
760 CALL CLEAR
770 CALL VCHAR(4,4,96,14)
780 CALL HCHAR(4,5,96,7)
790 CALL VCHAR(5,10,97)
800 CALL VCHAR(5,11,98)
810 REM ********************
815 REM CHOIX DU MOT ET AFFICHAGE DU NOM
BRE DE TIRETS
820 REM ********************
830 D=4
840 J=INT(RND*100)+1
850 M$=MOT$(J)
860 CALL HCHAR(10,16,95,LEN(M$))
870 REM ********************
880 REM CORPS DU JEU
890 REM ********************
900 DIM E$(30)
910 F=1
920 N=0
930 X=0 :: Y=0
940 DISPLAY AT(21,1):"PROPOSEZ UNE LETTR
E :"
950 CALL SOUND(100,2000,2)
960 CALL KEY(0,L,S)
970 IF S=0 THEN 960
980 L$=CHR$(L)
990 CALL VCHAR(21,25,L)
1000 FOR G=F TO 1 STEP -1
1010 IF L$=E$(G)THEN 2020
1020 NEXT G
1030 CALL VCHAR(D,31,L)
1040 F=F+1
1050 E$(F)=L$
1060 D=D+1
1070 R=1
1080 FOR X=1 TO LEN(M$)
1090 IF L$=SEG$(M$,X,1)THEN 1180
1100 NEXT X
1110 ON R GOTO 1120,1220
1120 DISPLAY AT(21,1):"PAS DE CHANCE"
1130 CALL SOUND(500,200,3)
1140 FOR I=1 TO 400
1150 NEXT I
1160 N=N+1
1170 ON N GOTO 1380,1460,1520,1580,1640,
1700,1790
1180 CALL VCHAR(10,15+X,L)
1190 Y=Y+1
1200 R=2
1210 IF Y=LEN(M$)THEN 1300 ELSE 1100
1220 DISPLAY AT(21,1):"BON CHOIX"
```

```
1230 CALL SOUND(500,1000,3)
1240 FOR I=1 TO 400
1250 NEXT I
1260 GOTO 940
1270 REM ******************
1280 REM FIN GAGNANTE
1290 REM ******************
1300 DISPLAY AT(21,1):"GAGNE !"
1310 CALL SOUND(400,784,3)
1320 CALL SOUND(400,1568,3)
1330 CALL SOUND(700,1319,3)
1340 GOTO 1910
1350 REM ******************
1360 REM TETE
1370 REM ******************
1380 CALL VCHAR(6,10,104)
1390 CALL VCHAR(7,10,105)
1400 CALL VCHAR(7,11,106)
1410 CALL VCHAR(6,11,107)
1420 GOTO 940
1430 REM ******************
1440 REM EPAULES
1450 REM ******************
1460 CALL VCHAR(8,10,108)
1470 CALL VCHAR(8,11,109)
1480 GOTO 940
1490 REM ******************
1500 REM BRAS DROIT
1510 REM ******************
1520 CALL VCHAR(8,9,110)
1530 CALL VCHAR(9,8,111)
1540 GOTO 940
1550 REM ******************
1560 REM BRAS GAUCHE
1570 REM ******************
1580 CALL VCHAR(8,12,112)
1590 CALL VCHAR(9,13,113)
1600 GOTO 940
1610 REM ******************
1620 REM CORPS
1630 REM ******************
1640 CALL VCHAR(9,10,114,3)
1650 CALL VCHAR(9,11,115,3)
1660 GOTO 940
1670 REM ******************
1680 REM JAMBE DROITE
1690 REM ******************
1700 CALL VCHAR(12,10,116)
1710 CALL VCHAR(13,10,117)
1720 CALL VCHAR(13,9,118)
1730 CALL VCHAR(14,9,116)
1740 CALL VCHAR(15,9,117)
1750 GOTO 940
1760 REM ******************
1770 REM JAMBE GAUCHE ET FIN
1780 REM ******************
1790 CALL VCHAR(12,11,119)
1800 CALL VCHAR(13,11,120)
1810 CALL VCHAR(13,12,121)
1820 CALL VCHAR(14,12,119)
1830 CALL VCHAR(15,12,120)
1840 CALL SOUND(200,247,3)
1850 CALL SOUND(200,247,3)
1860 CALL SOUND(200,247,3)
1870 CALL SOUND(800,196,2)
1880 DISPLAY AT(21,1):"PERDU !"
1890 DISPLAY AT(22,1):"LE MOT ETAIT"
1900 DISPLAY AT(22,14):M$
1910 DISPLAY AT(24,1):"ENCORE UNE PARTIE
? (O/N)"
1920 CALL KEY(0,C,E)
1930 IF E=0 THEN 1920
1940 IF C=78 THEN 1960
1950 IF C=79 THEN 760 ELSE 1920
1960 CALL CLEAR
1970 DISPLAY AT(15,10):"AU REVOIR"
1980 GOTO 2200
1990 REM ******************
2000 REM DEJA FAIT
2010 REM ******************
2020 DISPLAY AT(21,1):"LETTRE DEJA PROPO
SEE"
2030 CALL SOUND(500,3000,2)
2040 FOR H=1 TO 500
2050 NEXT H
2060 GOTO 940
2070 REM ******************
2080 REM LEXIQUE
2090 REM ******************
2100 DATA MAISON,JARDIN,LIVRE,JOYEUX,FIL
LETTE,VOITURE,FAUTEUIL,PAILLE,DOIGT,OREI
LLE
2110 DATA EXPLICATION,GYMNASTIQUE,KOALA,
WAGON,SYMPATIQUE,SYMPHONIE,OUIE,YAOURT,Q
UEUE,AMOUR
2120 DATA JAMBON,STYLO,POTENCE,BILLET,BA
LANCE,CAMION,CARAMEL,POISSON,GRENOUILLE,
BOUTEILLE
2130 DATA TABOURET,BALAYER,ACCENT,TRESOR
IER,MYSTERE,BOTILLON,AHURI,INOUI,BOURGEO
IS,FAUCILLE
2140 DATA ORDINATEUR,CRAYON,PANTALON,TAB
LE,CHAISE,ETAGE,BANQUETTE,CALCULATRICE,B
ANC,RIDEAU
2150 DATA DISQUE,DISCOURS,PARLOPHONE,DEN
TITION,ESPADRILLE,BRACELET,QUARTZ,MELANG
EUR,ROBINET,ETAMINE
2160 DATA ARMOIRE,FLEUR,POMME,LUNETTE,LU
MIERE,AMPHORE,ECRAN,TONNEAU,VIGNOBLE,BIE
RE
2170 DATA OREILLE,CHEVELURE,PEINTURE,LIB
RAIRIE,STYLO,MENOTTE,CHANDELLE,LUSTRERIE
,SERRURE,POUBELLE
2180 DATA KLAXON,VITRINE,MAGASIN,SOULIER
,CHEMIN,PROMENADE,CAMPAGNE,CIEL,JUSTICE,
CHAUSSETTE
2190 DATA RADIATEUR,ENCEINTE,INTERRUPTEU
R,TAILLEUR,SCHISTE,WHARF,ZERO,YACHT,MUSE
AU,ESTAMINET
2200 END
```

Fabriquez votre joystick

Robert Peltier

A première vue, construire soi-même un joystick ne présente pas un énorme intérêt : le matériel fourni par le constructeur n'est pas particulièrement ruineux et de plus il est bien adapté à sa fonction. Le montage proposé dans "99" aujourd'hui répond cependant à un besoin spécifique : permettre à un très jeune enfant d'accéder à l'ordinateur, sans pour autant compromettre la "survie" de ce dernier.

En effet, rares sont les enfants de cinq ans ou moins qui, face au clavier, se contentent d'effleurer les touches ou de manipuler avec calme le levier d'un joystick.

Tout au contraire, le comportement habituel de l'enfant est de rentrer dans le jeu avec toute la fraîcheur et la spontanéité de son âge... sans bien comprendre que ce jouet supporte moins bien les coups et les gestes brusques que son ours en peluche.

La première solution possible à ce problème est bien entendu de prendre l'enfant sur ses genoux et de tenter de canaliser ses gestes. Cette solution n'est pas à écarter tout-à-fait, mais est-il bien certain que l'enfant ait envie d'être en permanence sous surveillance ?

Une autre solution, plus radicale, consiste à interdire purement et simplement l'usage de l'ordinateur aux enfants qui ne possèdent pas encore un minimum de maîtrise d'eux-mêmes. Avouons qu'il serait triste de les priver ainsi de ce qu'ils considèrent, à juste titre, comme un merveilleux jouet : avant même de savoir lire ou écrire, un enfant peut parfaitement dessiner ou jouer de la musique, toutes choses qu'un ordinateur peut lui permettre de faire. Pourquoi, dans ces conditions, le priver d'un tel compagnon de jeu ?

Conçu avec cette arrière-pensée, le joystick "fait main" devait répondre aux exigences d'un "cahier des charges" très simple :

1 - le montage devait être fixé dans un boîtier métallique susceptible de résister à tous les chocs et à toutes les sollicitations auxquelles peut le soumettre un jeune enfant;

2 - pour des raisons évidentes de simplicité et de solidité, il ne devait comporter aucune pièce en mouvement, ce qui excluait la solution classique du "manche à balai";

3 - les fonctions du système devaient néanmoins permettre à l'enfant d'intervenir dans des programmes adaptés à son âge.

La solution adoptée consiste donc à regrouper sur la même face d'un boîtier de montage 4 interrupteurs. Selon le programme utilisé, le système fonctionne soit en déterminant 8 directions (comme avec un joystick classique), soit en fournissant à l'enfant 4 choix d'intervention dans le déroulement d'un programme, suivant les fonctions attribuées à chacune des touches.

Le principe de fonctionnement est très simple : comme l'indiquent les instructions d'installation du TI-99, la prise à 9 broches destinée aux télécommandes se trouve sur le côté gauche de l'appareil. Sa forme est celle d'un trapèze dont la base la plus large est orientée vers le haut. On remarque (à condition d'avoir de bons yeux) que ces broches sont numérotées de 1 à 9 (figure 1).

Pour déterminer la fonction de ces broches, on lance le programme suivant (après avoir déverrouillé la touche ALPHA-LOCK) :

```
100 CALL JOYST(1,A,B)
110 IF (A= 0)★ (B= 0) THEN 100
120 PRINT A;B
130 GOTO 100
```

Puis, à l'aide d'un fil électrique de quelques centimètres, on met en contact les broches deux à deux, en observant les valeurs affichées à l'écran. On obtient ainsi les données du tableau suivant :

	A	B
Aucun contact	0	0
Contact broches 3-7	0	4
" 8-7	0	− 4
" 5-7	− 4	0
" 9-7	4	0

```
50 REM ====================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET ROBERT PELTIER
80 REM ====================
90 REM
100 CALL CLEAR
110 PRINT "RELEVEZ LA TOUCHE"
115 PRINT
120 PRINT "alpha lock"
125 PRINT
130 PRINT "ET ENFONCEZ UNE TOUCHE"
135 PRINT
140 CALL KEY(0,K,S)
150 IF S=0 THEN 140
160 CALL CHAR(96,"FFFFFFFFFFFFFFFF")
170 CALL CLEAR
180 X=10
190 Y=10
200 CALL SCREEN(2)
210 CALL COLOR(2,7,8)
220 CALL COLOR(9,7,2)
230 CALL JOYST(1,A,B)
240 CALL KEY(0,K,S)
250 IF K=32 THEN 170
260 IF X<1 THEN 270 ELSE 280
270 X=X+1
280 IF X>24 THEN 290 ELSE 300
290 X=X-1
300 IF Y<1 THEN 310 ELSE 320
310 Y=Y+1
320 IF Y>32 THEN 330 ELSE 340
330 Y=Y-1
340 CALL HCHAR(X,Y,96)
350 CALL HCHAR(X,Y,42)
360 IF B=-4 THEN 370 ELSE 380
370 X=X+1
380 IF B=4 THEN 390 ELSE 400
390 X=X-1
400 IF A=-4 THEN 410 ELSE 420
410 Y=Y-1
420 IF A=4 THEN 430 ELSE 230
430 Y=Y+1
440 GOTO 230
```

La broche 7 sert, ainsi que vous le voyez, exclusivement aux branchements du joystick numéro 1. D'autres combinaisons, de structure similaire, sont utilisées pour le joystick numéro 2. Nous vous laissons le soin de les découvrir

En remplaçant la ligne 100 par 100 CALL JOYST(2,A,B), il est également possible de déterminer les bornes à utiliser pour le montage d'un deuxième joystick.

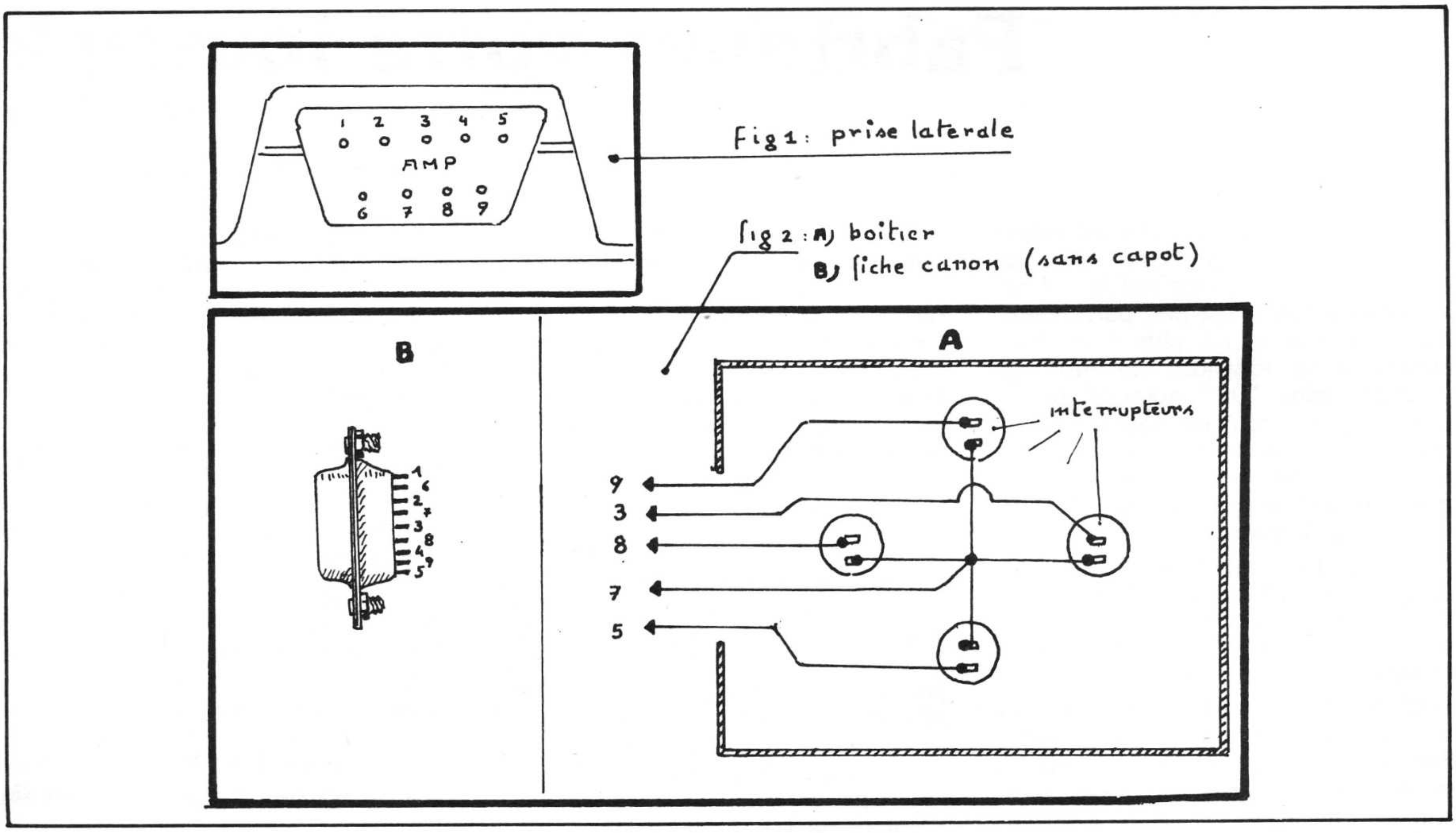
La réalisation de notre joystick ne nécessite donc que la mise en place d'interrupteurs dans un boîtier et la liaison du système avec l'ordinateur suivant le schéma de la figure 2. La seule difficulté rencontrée est de percer les 5 trous dans le boîtier (4 pour les interrupteurs +1 pour le câble de liaison).

Plus délicat est le problème de la liaison de ce montage avec la prise de télécommande : celle-ci se fait à l'aide d'un cordon spirale et d'une prise Canon DB, elle-même logée dans un capot : la taille réduite des broches de la prise Canon nécessite l'emploi d'un fer à souder à panne fine et... beaucoup de patience.

Ce joystick n'est certainement pas aussi précis qu'un joystick muni du classique "manche à balai", mais il est cependant possible de déterminer grâce à lui les 8 directions du déplacement d'un mobile sur l'écran en agissant soit sur un seul interrupteur, soit sur deux interrupteurs contigus. Le programme de la figure 3 permet de vérifier le bon fonctionnement du joystick.

NDLR : Dans un premier temps, Robert Peltier n'est pas vendeur de son joystick. A vous donc de le bricoler !

Voir schéma page suivante



Appelez-moi KEY

Bernard Braun

Dans le dialogue homme-machine, ce qu'affiche l'ordinateur est, par définition, correct. Les réponses de l'opérateur, par contre, sont beaucoup plus sujettes à erreur et rendent de ce fait les programmes vulnérables. Pour obtenir une information, les programmes les plus simples utilisent l'instruction INPUT ; cependant cette macro-instruction n'est absolument pas protégée contre les erreurs de l'utilisateur. Son premier handicap réside dans le fait que cette instruction manifeste son indignation par des messages dont l'interprétation est réservée au seul initié. Citons quelques exemples tirés du bêtisier le plus courant :

10 INPUT "ADRESSE" :AD$
Réponse de l'opérateur :
22, RUE DES ALLONGES (ENTER)
Diagnostic de la machine :
INPUT ERROR IN 10
TRY AGAIN :

40 INPUT "PRIX UNITAIRE" :PU
Réponse de l'opérateur :
9.90 F (ENTER)
Diagnostic de la machine :
INPUT ERROR IN 40
TRY AGAIN :

60 INPUT "NOTE SUR " :NO
Réponse de l'opérateur :
(ENTER)
Diagnostic de la machine :
INPUT ERROR IN 60
TRY AGAIN :

Le deuxième handicap de l'instruction INPUT est sa totale impuissance à transmettre les caractères de contrôle qui peuvent donner de la souplesse au dialogue.
Pour éviter les limitations de l'instruction INPUT, les micro-ordinateurs évolués disposent d'une instruction élémentaire qui déchiffre un par un chaque caractère tapé au clavier pour en traiter l'information selon les besoins du programme. Sur le TI-99/4A, cette instruction est le sous-programme appelé KEY ; ailleurs, ce sera l'ordre GET, ou la pseudo-variable INKEY$.

D'une très grande souplesse d'utilisation, le sous-programme KEY utilise un argument d'entrée (le mode d'utilisation du clavier), et deux variables de sortie (le code de la touche enfoncée et une variable d'état indiquant si une touche est actionnée ou reste enfoncée). La syntaxe d'emploi du sous-programme KEY est la suivante : CALL KEY(mode clavier, code touche, variable d'état). Afin de faire preuve d'originalité dans les exemples utilisés, le mode clavier, lorsqu'il n'est pas explicite, sera appelé MODE ; le code de la touche sera renvoyé sur la variable CODE, KEY ou simplement K ; et la variable d'état sera nommée STATUS ou S.
Le programme 1 ci-après fait apparaître les variations de la variable d'état STATUS ; notons que si aucune touche n'est actionnée, le code retourné par le sous-programme KEY est −1 et n'est donc pas un code ASCII. Le programme 2 illustre les utilisations classiques de la variable d'état lors d'une entrée par clavier.

Dans un premier cas, on entre TEST=−1 et le programme écrit des −1 sur l'écran tant qu'une touche est appuyée.
Pour TEST=0, il y a écriture de zéros sur l'écran seulement quand une touche est enfoncée.
Enfin, si on entre TEST=1, la machine écrit un 1 unique lors de la pression sur une touche.

Le clavier du TI-99/4A, très sophistiqué, peut être configuré de 4 façons différentes selon les valeurs de l'argument MODE.
Les modes 1 et 2 servent à diviser le clavier de la console en deux moitiés,

```
10 CALL KEY(0,CODE,STATUS)
20 PRINT "STATUS: ";STATUS;"   CODE: ";CODE
30 GOTO 10
40 END
```

Programme 1

```
10 INPUT "TEST STATUS ":TEST
20 PRINT "STATUS = ";
30 CALL KEY(0,KEY,STATUS)
40 IF STATUS<>TEST THEN 30
50 PRINT TAB(10);STATUS
60 GOTO 30
```

Programme 2

ce qui est utile pour certains jeux. L'argument de clavier 3 est automatiquement choisi au départ (valeur par défaut) par le TI Basic, la touche CTRL étant alors inactive. Avec les arguments 4 et 5, toutes les touches sont actives; les codes des caractères de contrôle sont compris entre 1 et 31 dans le premier cas, entre 129 et 159 dans le second. Notons que le mode 4 présente l'avantage de configurer un clavier compatible avec des claviers de grande diffusion comme celui de l'Apple II, et que, dans ce mode, ENTER a toujours le code 13, *contrairement à ce qu'affirme le manuel d'utilisation.*

La déclaration d'un mode de clavier nul dans CALL KEY ne définit pas une nouvelle configuration, mais conserve la configuration précédente, ce qui économise le temps d'une réaffectation à chaque appel du sous-programme.

Le troisième programme de démonstration affiche, selon la configuration de clavier choisie, le code d'une touche pendant qu'elle est enfoncée.

Les programmes élémentaires 1, 2 et 3 ont servi à disséquer le fonctionnement du sous-programme KEY. Pour montrer ce sous-programme en situation, voici une "subroutine" en vraie grandeur tirée d'un programme de calcul des moyennes de notes dans une classe.

Le programme 4 n'est pas conçu pour être autonome. Pour l'expérimenter, il est bon de l'insérer dans un programme tel que notre exemple 4bis, programme qui définit une fois pour toutes la configuration du clavier en mode 4 et exploite les paramètres renvoyés par la subroutine.

Pour paraître dans le vent, de nombreux périodiques dans les spécialités les plus variées publient des programmes en Basic. Ces programmes sont intéressants et leur fonctionnement garanti, mais ils sont toujours spécifiques d'une machine particulière. Le sport à la mode devient la transcription d'un programme quelconque pour le rendre compatible à un autre système. Dans cet esprit, il est intéressant de comparer les implantations des trois procédures parallèles GET, INKEY$ et CALL KEY.

La syntaxe de l'instruction GET est extrêmement simple : GET(nom de variable) affecte à la variable mentionnée la première valeur lue au clavier, en général un caractère. Certains appareils acceptent aussi des variables numériques, mais nous n'envisagerons pas ce cas.
Le fonctionnement de l'instruction diffère légèrement selon la technique employée pour la scrutation du clavier et, à ce titre, la comparaison des systèmes Commodore et Apple II est significative. Le clavier des CBM est lu régulièrement soixante fois par seconde, si bien qu'un GET A$ a toutes les chances d'affecter à la variable A$ la valeur CHR$(0) qui ne correspond à aucune touche enfoncée. Pour traduire exactement une telle instruction par un CALL KEY, il faut quatre lignes :

100 CALL KEY(0,KEY,STATUS)
101 IF STATUS=1 THEN 103
102 LET KEY=0
103 LET A$=CHR$(KEY)

Cette traduction, pour intéressante qu'elle soit, est irréaliste car, dans la quasi-intégralité des programmes, le caractère nul est éliminé par un test. On trouve alors les montages équivalents :

1 - GET :
100 GET A$
110 IF A$="" GOTO 100

2 - KEY :
100 CALL KEY(0,KEY,STATUS)
101 IF STATUS<>1 THEN 100
102 LET A$=CHR$(KEY)

Le test de la ligne 110 est inutile en TI-Basic.

Le clavier de l'Apple II, lui, génère un signal "strobe" à chaque action sur une touche. Ce signal est détecté par l'instruction GET elle-même pour bloquer le programme tant qu'une nouvelle touche n'est pas enfoncée. Les trois lignes 100, 101 et 102 correspondent à une ligne unique en Basic Applesoft :

100 GET A$

CALL KEY et INKEY$

Dans de nombreux systèmes à base de Z80, comme le Sinclair ZX81, INKEY$ est une variable dont la valeur, déterminée par le système, est le caractère nul quand aucune touche n'est enfoncée, et le caractère de la touche actionnée dans le cas contraire. L'utilisation classique de cette pseudo-variable tient en deux lignes dont la transcription en CALL KEY prend trois lignes :

1 - INKEY$:
100 LET A$=INKEY$
110 IF A$="" THEN GOTO 100

```
100 REM *CALL KEY DEMO* / ENTREE DE NOTE SUR
 VINGT
110 REM ENNA PARIS NORD / DEPARTEMENT MATHEM
ATIQUE
120 REM
130 REM VARIABLES INTERNES: FLAP, KEY, STATU
S
140 REM VARIABLE EN ENTREE: NEANT
150 REM VARIABLE EN SORTIE: FLAG, NOTE$
160 REM ADRESSE D'ENTREE: 200, MODE CLAVIER:
 4
170 REM
180 CALL SOUND(300,196,0)
190 PRINT "  NOTE ANNULEE"
200 LET FLAG=0
210 LET FLAP=0
220 LET NOTE$="0"
230 PRINT "    > ";
240 CALL KEY(0,KEY,STATUS)
250 IF STATUS<>1 THEN 240
260 IF (KEY>47)*(KEY<59)THEN 370
270 IF KEY=46 THEN 350
280 IF KEY=13 THEN 400
290 IF (KEY=2)+(KEY>3)THEN 320
300 LET FLAG=KEY
310 GOTO 400
320 IF KEY=24 THEN 180
330 IF KEY<>44 THEN 240
340 LET KEY=46
350 IF FLAP=1 THEN 240
360 LET FLAP=1
370 LET NOTE$=NOTE$&CHR$(KEY)
380 PRINT CHR$(KEY);
390 GOTO 240
400 IF VAL(NOTE$)>20 THEN 180
410 RETURN
```

Programme 4

```
10 INPUT "MODE ":MODE
20 CALL CLEAR
30 CALL KEY(MODE,KEY,STATUS)
40 IF STATUS=0 THEN 30
50 PRINT "MODE: ";MODE,"  KEY: ";KEY
60 CALL KEY(0,K,S)
70 IF S=0 THEN 20 ELSE 60
80 END
```

Programme 3

```
10 GOTO 460
100 REM ICI PREND PLACE LE
300 REM SOUS PROGRAMME P.4
410 REM D'ENTREE DE NOTE
460 CALL KEY(4,KEY,STATUS)
470 PRINT
480 PRINT "TAPEZ UNE NOTE, PUIS ENTER"
490 PRINT "        CONTROL A POUR ABSENT"
500 PRINT "        CONTROL C POUR CESSER"
510 GOSUB 200
520 PRINT
530 IF FLAG<>0 THEN 560
540 PRINT "VOUS AVEZ TAPE LA NOTE";VAL(NOTE$

550 GOTO 470
560 IF FLAG=3 THEN 590
570 PRINT "ELEVE ABSENT"
580 GOTO 470
590 PRINT "*** FIN ***"
600 END
```

Programme 4 b

2 - CALL KEY :

```
100 CALL KEY(0,KEY,STATUS)
110 IF STATUS=0 THEN 100
120 LET A$=CHR$(KEY)
```

Ces différents exemples de la transcription au plus près de l'instruction GET ou de la pseudo-variable INKEY$ par un sous-programme KEY sont systématiquement défavorables à ce dernier. Cela est normal : la traduction littérale d'un programme Basic d'un dialecte dans un autre ne peut qu'alourdir ce programme. Pour comparer la souplesse et le volume des trois procédures présentées ici, il est plus sage de les juger en situation dans quatre études de cas.

Premier cas : arrêt du programme tant qu'aucune touche n'est actionnée.

1 - GET :

```
500 GET A$
510 IF A$="" GOTO 500
```

2 - INKEY$:

```
500 IF INKEY$="" THEN
    GOTO 500
```

3 - CALL KEY :

```
500 CALL KEY(0,K,S)
510 IF S=0 THEN 500
```

Deuxième cas : attente du retour chariot (ENTER ou NEWLINE).

1 - GET :

```
800 GET A$
810 IF A$ CHR$(13) GOTO 800
```

2 - INKEY$:

```
800 IF INKEY$ CHR (118) then 800
```

(notons que le ZX81 n'utilise pas les codes ASCII)

3 - CALL KEY :

```
800 CALL KEY(0,K,S)
810 IF KEY=13 THEN 800
```

Troisième cas : choix multiple (A, B, C et D par exemple)

1 - GET :

```
900 GET A$
910 IF A$="A" OR A$="D"
    THEN 900
920 ON ASC(A$)-64
    GOTO 100,110,120,130
```

2 - INKEY$:

```
900 LET A$=INKEY$
910 IF A$="A" OR A$="D"
    THEN 900
920 ON ASC(A$)-64
    GOTO 100,110,120,130
```

3 - CALL KEY :

```
900 CALL KEY(0,K,S)
910 IF (K 65)+(K 68) THEN 900
920 ON K-64 GOTO 100,110,
    120,130
```

Quatrième cas : entrée de caractères.

1 - GET :

```
200 GET A$
210 IF A$="" THEN 200
..................
300 PRINT A$
310 GOTO 200
```

2 - INKEY$:

```
200 IF INKEY$="" THEN 200
210 LET A$=INKEY$
..................
300 PRINT A$
310 GOTO 200
```

3 - CALL KEY :

```
200 CALL KEY(0,K,S)
210 IF S<1 THEN 200
..................
300 PRINT CHR$(K)
310 GOTO 200
```

Exception faite du choix multiple en INKEY$, les procédures citées sont extraites de programmes tournant sur CBM 3000 et 4000, ITT 2020, Apple II, ZX81, ou TI-99/4A. Ces procédures n'ont pas été sélectionnées pour leur généralité ou leur diffusion, mais à cause du caractère exemplaire qu'elles présentent.

En guise de conclusion, nous pourrions suggérer à tout utilisateur potentiel d'un sous-programme KEY de répondre d'abord aux questions suivantes :

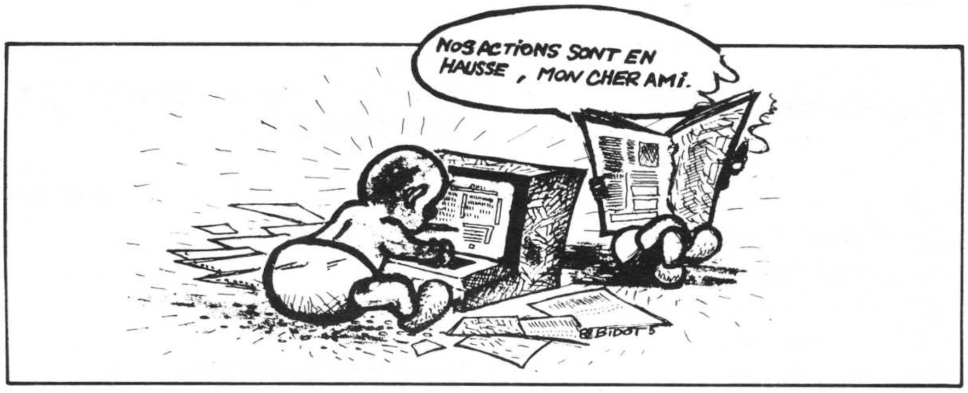
- le programme attend-il l'état d'une touche ou l'action sur une touche (transition) ? Dans le premier cas, le test de la variable STATUS peut être remplacé par une test de signe sur la variable KEY.
- la valeur de la touche concernée est-elle importante ? Dans la négative, le test de la variable KEY peut être omis.
- dans l'affirmative, cette valeur intervient-elle sous forme d'un caractère ou sous forme numérique ?

```
50 REM ====================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET BERNARD BRAUN
80 REM ====================
90 REM
100 CALL SCREEN(8)
110 GOTO 440
120 REM *CALL KEY DEMO* / ENTREE DE NOTE SUR
 VINGT
130 REM ENNA PARIS NORD / DEPARTEMENT MATHEM
ATIQUE
140 REM
150 REM VARIABLES INTERNES: FLAP, KEY, STATU
S
160 REM VARIABLE EN ENTREE: NEANT
170 REM VARIABLE EN SORTIE: FLAG, NOTE$
180 REM ADRESSE D'ENTREE: 200, MODE CLAVIER:
 4
190 REM
200 CALL SOUND(300,196,0)
210 PRINT "  NOTE ANNULEE"
220 LET FLAG=0
230 LET FLAP=0
240 LET NOTE$="0"
250 PRINT "    >  ";
260 CALL KEY(0,KEY,STATUS)
270 IF STATUS<>1 THEN 260
280 IF (KEY>47)*(KEY<59)THEN 390
290 IF KEY=46 THEN 370
300 IF KEY=13 THEN 420
310 IF (KEY=2)+(KEY>3)THEN 340
320 LET FLAG=KEY
330 GOTO 420
340 IF KEY=24 THEN 200
350 IF KEY<>44 THEN 260
360 LET KEY=46
370 IF FLAP=1 THEN 260
380 LET FLAP=1
390 LET NOTE$=NOTE$&CHR$(KEY)
400 PRINT CHR$(KEY);
410 GOTO 260
420 IF VAL(NOTE$)>20 THEN 200
430 RETURN
440 CALL KEY(4,KEY,STATUS)
450 PRINT
460 PRINT "TAPEZ UNE NOTE, PUIS ENTER"
470 PRINT "        CONTROL A POUR ABSENT"
480 PRINT "        CONTROL C POUR CESSER"
490 GOSUB 220
500 PRINT
510 IF FLAG<>0 THEN 540
520 PRINT "VOUS AVEZ TAPE LA NOTE";VAL(NOTE$
)
530 GOTO 450
540 IF FLAG=3 THEN 570
550 PRINT "ELEVE ABSENT"
560 GOTO 450
570 PRINT "*** FIN ***"
580 END
```

Programme 4 + 4 b

Programmes d'aide à la décision

Didier Bicking



La sollicitation d'un emprunt est une situation à laquelle tout le monde est confronté un jour ou l'autre. Cependant, très peu de gens savent comment en sont calculées les mensualités de remboursement. J'ai découvert, un jour, la formule permettant de calculer la mensualité pour un emprunt à remboursements constants. J'en ai profité pour construire un programme autour de cette formule, en faisant en sorte que cette construction puisse servir d'exemple pour d'autres applications (à vous de les imaginer!)

Principe du programme

Le programme, tel que vous pouvez le voir, est écrit en Basic Etendu : j'ai en effet utilisé au maximum les instructions DISPLAY AT et ACCEPT AT, disponibles seulement dans ce langage. Le principe de base en est le suivant :

1 - le manipulateur indique les 4 valeurs servant de base au calcul de la mensualité : montant de l'emprunt, taux hors assurance, taux d'assurance et durée de remboursement. ACCEPT AT permet de diriger son action directement là où elle est nécessaire et nulle part ailleurs.

2 - des calculs intermédiaires sont faits sur les taux et affichés sur l'écran comme faisant partie intégrale des données de base.

3 - le montant de la mensualité est calculé à la ligne 330. Remarquez à ce sujet la structure de la formule : INT(100*valeur+0.5)/100 permet d'obtenir un montant arrondi au centime le plus proche.

4 - la ligne 340 réalise l'affichage de ce montant. Le USING qu'elle comporte formate cet affichage en un montant à deux décimales. Ainsi, même les zéros non significatifs seront présents ce qui, dans le cas d'un montant en francs et centimes, est quand-même plus agréable.

5 - de la même manière, la ligne 350 calcule et affiche le montant total des remboursements.

6 - le manipulateur peut ensuite modifier l'une des données. Aussitôt qu'il a procédé à cette modification, l'ensemble des calculs est refait et la possibilité lui est offerte de faire de nouvelles modifications.

Pour obtenir cette dernière caractéristique du programme, j'ai disposé les instructions de rentrée des données dans un ensemble de sous-programmes (lignes 470 à 500). Ces sous-programmes sont appelés une première fois à partir de la ligne 250 pour la rentrée des données de départ, puis une deuxième fois à partir de la ligne 420 pour opérer les modifications.

Remarquez que toutes les instructions ACCEPT AT comportent un VALIDATE. Les lignes 480 à 500 acceptent, en plus des chiffres, les autres signes numériques dont notamment le point décimal. La ligne 510 demande le nombre de mensualités. Il est donc hors de question d'y voir figurer autre chose qu'un nombre entier et le VALIDATE(DIGIT) n'autorisera pas la rentrée d'un point décimal.

Extensions possibles de ce programme

La première extension envisageable est, bien entendu, de faire imprimer par le programme ce qui apparaît sur l'écran, du moins pour ceux qui disposent d'une imprimante...

Ce programme peut avoir deux types d'utilisation inverses consistant à rechercher par tâtonnements le montant que vous pouvez emprunter en fonction de votre capacité d'endettement, ou le taux d'un crédit qui vous est proposé avec pour seules indications les montant, durée et mensualités de remboursement. Il vous faut alors remplir les trois rubriques dont les montants sont connus, et faire des essais successifs pour la valeur de la dernière rubrique jusqu'à l'atteinte du résultat souhaité.

Cette structure de programme peut être utilisée pour de nombreuses applications, notamment dans le domaine des outils d'aide à la décision. Le principe est le suivant :

1 - un certain nombre de valeurs présentent entre elles des relations mathématiques précises ;

2 - toutes ces relations aboutissent à la composition d'un résultat final ;

3 - le problème est de savoir quelle est l'influence sur le résultat de la variation d'une ou plusieurs de ces valeurs de base.

Le principe de base des célèbres programmes Visicalc (de la société Visicorp) et Multiplan (de la société Microsoft) n'est rien d'autre que cela! Voici d'ailleurs un second programme qui illustre mon propos. Il s'agit cette fois de l'établissement d'un compte d'exploitation prévisionnel pour une PME.

Compte d'exploitation prévisionnel

Le fonctionnement de ce programme est le suivant :

- on entre successivement les valeurs des 9 rubriques de débit du compte, puis des 9 rubriques de crédit;
- un tableau est ensuite présenté, faisant apparaître le résultat. Ce résultat est présenté de manière comptable : un bénéfice est écrit au débit, une perte au crédit, de sorte que les totaux de colonnes s'équilibrent;
- Il est alors possible de modifier individuellement chacune des rubriques, au débit ou au crédit. Après chaque modification, le compte est entièrement recalculé. On peut alors juger directement du résultat de chaque décision de stratégie de l'entreprise, et ajuster ces décisions en fonction du résultat recherché.

De nombreux programmes peuvent être construits selon un modèle identique. Ils permettront à leur utilisateur de prendre, en parfaite connaissance de cause, la bonne décision au bon moment.

Programme "Prêt"

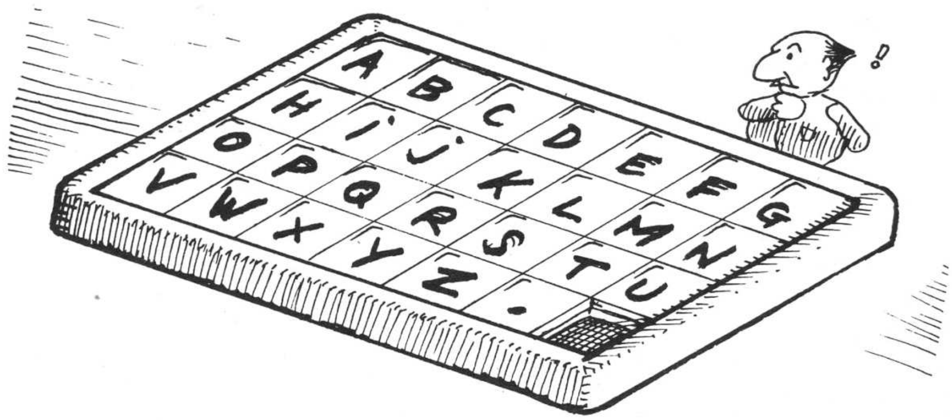
```
50 REM =====================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET DIDIER BICKING
80 REM =====================
100 CALL CLEAR :: CALL SCREEN(14)
110 REM *******************
120 REM PREPARATION DE L'ECRAN
130 REM *******************
140 DISPLAY AT(1,2):"EMPRUNT A REMBOURSEMENT
S" :: DISPLAY AT(2,7):"CONSTANTS"
150 DISPLAY AT(3,0):"-----------------------
-----"
160 DISPLAY AT(4,1):"1-MONTANT     :"
170 DISPLAY AT(6,1):"2-TAUX HORS AS:" :: DIS
PLAY AT(7,3):"soit mensuel:"
180 DISPLAY AT(9,1):"3-TAUX ASSUR. :" :: DIS
PLAY AT(10,3):"soit mensuel:"
190 DISPLAY AT(12,3):"TAUX TOTAL  :" :: DISP
LAY AT(13,3):"soit mensuel:"
200 DISPLAY AT(15,1):"4-NOMBRE DE" :: DISPLA
Y AT(16,3):"MENSUALITES :"
210 DISPLAY AT(17,0):"----------------------
------"
220 REM *******************
230 REM APPEL DES SOUS-PROGRAMMES DE RENTREE
 DES DONNEES
240 REM *******************
250 GOSUB 480 :: GOSUB 490 :: GOSUB 500 :: G
OSUB 510
260 REM *******************
270 REM CALCULS ET AFFICHAGES
280 REM *******************
290 TMB=TB/1200 :: DISPLAY AT(7,17):TMB
300 TMAS=TAS/1200 :: DISPLAY AT(10,17):TMAS
310 TT=TB+TAS :: DISPLAY AT(12,17):TT
320 TMT=TT/1200 :: DISPLAY AT(13,17):TMT
330 RM=INT(100*MONT*TMT/(1-((1+TMT)^-MEN))+0
.5)/100 :: DISPLAY AT(18,3):"MONTANT
 DES" :: DISPLAY AT(19,3):"MENSUALITES :"
340 DISPLAY AT(19,17):USING "######.##":RM
350 TR=RM*MEN :: DISPLAY AT(21,3):"SOIT TOTA
L  :" :: DISPLAY AT(21,16):USING "##
####.##":TR
360 DISPLAY AT(22,0):"----------------------
------"
370 REM *******************
380 REM MODIFICATION PAR RUBRIQUE
390 REM *******************
400 DISPLAY AT(23,2):"RUBRIQUE A CHANGER : "
410 ACCEPT AT(23,23)VALIDATE(DIGIT)BEEP:A
420 IF A<1 OR A>4 THEN END
430 ON A GOSUB 480,490,500,510
440 GOTO 290
450 REM *******************
460 REM RENTREE DES DONNEES
470 REM *******************
480 ACCEPT AT(4,17)VALIDATE(NUMERIC)BEEP:MON
T :: RETURN
490 ACCEPT AT(6,17)VALIDATE(NUMERIC)BEEP:TB
:: RETURN
500 ACCEPT AT(9,17)VALIDATE(NUMERIC)BEEP:TAS
 :: RETURN
510 ACCEPT AT(16,17)VALIDATE(DIGIT)BEEP:MEN
:: RETURN
```

Programme "CEP"

```
50 REM =====================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET DIDIER BICKING
80 REM =====================
100 REM *******************
110 REM INITIALISATIONS
120 REM *******************
130 CALL CLEAR :: CALL SCREEN(14)
140 TD=0 :: TC=0 :: R=0
150 REM ******************
160 REM RUBRIQUES DEBIT
170 REM ******************
180 DISPLAY AT(1,3):"RUBRIQUES DEBIT :"
190 DISPLAY AT(2,1):"-----------------------
----"
200 FOR I=1 TO 9
210 DISPLAY AT(I+2,2):"RUBRIQUE";I
220 ACCEPT AT(I+2,16)VALIDATE(NUMERIC)BEEP:R
D(I)
230 TD=TD+RD(I)
240 NEXT I
250 DISPLAY AT(12,1):"----------------------
-----"
260 DISPLAY AT(13,2):"TOTAL" :: DISPLAY AT(1
3,16):USING "######.##":TD
270 DISPLAY AT(16,2):"UNE TOUCHE POUR LA SUI
TE"
280 CALL KEY(0,K,S):: IF S=0 THEN 280
290 CALL CLEAR
300 REM ******************
310 REM RUBRIQUES CREDIT
320 REM ******************
330 DISPLAY AT(1,3):"RUBRIQUES CREDIT :"
340 DISPLAY AT(2,1):"-----------------------
----"
350 FOR I=1 TO 9
360 DISPLAY AT(I+2,2):"RUBRIQUE";I :: ACCEPT
 AT(I+2,16)VALIDATE(NUMERIC)BEEP:RC(
I)
370 TC=TC+RC(I)
380 NEXT I
390 DISPLAY AT(12,1):"----------------------
-----"
400 DISPLAY AT(13,2):"TOTAL" :: DISPLAY AT(1
3,16):USING "######.##":TC
410 DISPLAY AT(16,2):"UNE TOUCHE POUR LA SUI
TE"
420 CALL KEY(0,K,S):: IF S=0 THEN 420
430 CALL CLEAR :: CALL SCREEN(4)
440 REM ******************
450 REM PRESENTATION DU COMPTE
460 REM ******************
470 DISPLAY AT(1,5):"DEBIT        ù   CREDIT"
480 DISPLAY AT(2,1):"-----------------------
----"
490 FOR I=1 TO 9
500 DISPLAY AT(I+2,2):I :: DISPLAY AT(I+2,6)
:USING "######.##":RD(I):: DISPLAY A
T(I+2,17):"ù"
510 DISPLAY AT(I+2,20):USING "######.##":RC(
I)
520 NEXT I
530 DISPLAY AT(12,1):"----------------------
-----"
540 REM ******************
550 REM PRESENTATION DU SOLDE
560 REM ******************
570 R=TC-TD
580 IF R>0 THEN DISPLAY AT(13,6):USING "####
##.##":R :: DISPLAY AT(13,20):"
    " :: TD=TD+R :: GOTO 600
590 DISPLAY AT(13,6):"          " :: DISPLAY
AT(13,20):USING "######.##":ABS(R)::
 TC=TC+ABS(R)
600 DISPLAY AT(14,1):"----------------------
-----"
610 DISPLAY AT(15,6):USING "######.##":TD ::
 DISPLAY AT(15,20):USING "######.##"
:TC
620 DISPLAY AT(16,1):"----------------------
-----"
630 REM ******************
640 REM PROCEDURE DE MODIFICATION
650 REM ******************
660 DISPLAY AT(18,2):"VOULEZ-VOUS MODIFIER(D
,C)? " :: ACCEPT AT(18,28)VALIDATE("
CD")BEEP:A$
670 DISPLAY AT(20,2):"QUELLE RUBRIQUE ? " ::
 ACCEPT AT(20,28)VALIDATE(DIGIT)BEEP
:I
680 IF A$="C" THEN 740
690 IF R>0 THEN LET TD=TD-RD(I)-R :: GOTO 71
0
700 TD=TD-RD(I):: TC=TC+R
710 ACCEPT AT(I+2,6)VALIDATE(NUMERIC)BEEP:RD
(I)
720 TD=TD+RD(I)
730 GOTO 490
740 IF R<0 THEN LET TC=TC-RC(I)+R :: GOTO 76
0
750 TC=TC-RC(I):: TD=TD-R
760 ACCEPT AT(I+2,20)VALIDATE(NUMERIC)BEEP:R
C(I)
770 TC=TC+RC(I)
780 GOTO 490
```

Le jeu de l'Alphabet

Denise Amrouche



Ce jeu est lui aussi la transcription pour ordinateur d'un jeu bien connu : nous avons tous eu, un jour ou l'autre, en main l'un de ces casse-têtes composé d'un cadre dans lequel évoulent 27 caractères et une case vide. Il s'agit de remettre les caractères dans l'ordre, sans bien entendu démonter le jeu pour le remonter ensuite (j'en connais qui le faisaient !)

Dans cette version pour TI-99, les caractères sont les 26 lettres de l'alphabet et un point. L'ordinateur les dispose au hasard dans un cadre de 7⋆4, laissant la case vide en bas et à droite. Pour déplacer un caractère, le joueur n'a qu'à appuyer sur la touche correspondante du clavier.

Le programme débute par la constitution des formes et des couleurs. Ainsi, l'écran sera magenta (ligne 160) alors que les caractères seront noirs sur fond jaune (lignes 180 à 210). Le cadre du jeu sera composé d'un rectangle de deux traits (lignes 230 à 340) de couleur jaune clair (ligne 170).

La phase d'initialisation consiste à disposer les caractères de façon aléatoire dans le cadre. Le programme commence par les ranger en ordre alphabétique dans un tableau B en les identifiant par leurs codes ASCII (lignes 380 à 420). Ensuite,il choisit aléatoirement l'un de ces codes (lignes 440 à 460) et dispose le caractère correspondant dans la première case (ligne 490). Enfin, il élimine ce code du tableau B (lignes 500 et 510) et recommence la même procédure pour la case suivante.

Un procédé différent aurait pu être utilisé pour cette phase d'initialisation. Il consisterait à tirer les lettres dans l'ordre alphabétique et à les disposer dans des cases choisies de manière aléatoire. Mais cela nous aurait contraint à vérifier à chaque fois que la case choisie n'est pas déjà occupée et cette vérification aurait été d'autant plus longue que l'on approcherait de la fin de l'alphabet.

Les lignes 570 à 670 vérifient que les lettres et le point se trouvent dans l'ordre (le point devant être disposé derrière le Z). Si ce n'est pas le cas, le programme renvoie à la procédure d'attente des ordres à exécuter.

L'exécution des ordres se fait dans les lignes 680 à 980. Cette série d'instructions commence par deux CALL KEY (lignes 710 et 730). C'est là une sage précaution car, sans les lignes 730 et 740, si on laisse le doigt trop longtemps sur une touche, on risque de voir le caractère désigné se déplacer vers la case vide, puis revenir aussitôt à la case qu'il occupait précédemment.

La ligne 710 détermine la valeur de la variable CHOIX, désignant le caractère à déplacer. La ligne 730, elle, donnera à la variable S la valeur −1 tant que la touche restera enfoncée. Si cette valeur reste à −1, la ligne 740 renvoie le programme à la ligne 730, et ainsi de suite jusqu'à ce que la touche ait été relâchée. Il n'y aura ainsi de valse hésitation de la part d'un caractère que si, volontairement, le joueur a appuyé deux fois sur la même touche.

Le reste de la procédure d'exécution des ordres consiste, bien entendu, à vérifier que le caractère désigné peut être déplacé (lignes 750 à 940). Si ce n'est pas le cas, le programme renvoie à la ligne 710 (qui demande un nouvel ordre). Sinon, l'ordre est exécuté par les lignes 950 à 970.

Après chaque ordre, le programme retourne à la ligne 600 pour examiner si l'ordre voulu a été atteint, auquel cas le joueur reçoit les félicitations de l'ordinateur.

Les règles de ce jeu étant particulièrement simples, il ne semble nécessaire d'y apporter aucune modification. Seul le graphisme peut être modifié selon votre goût. Surtout ne vous en privez pas si les dessins qui vous sont présentés ne correspondent pas exactement à la façon dont vous auriez souhaité que ce jeu soit illustré. Et amusez-vous bien !

```
50 REM ====================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET DENISE AMROUCHE
80 REM ====================
90 REM
100 REM JEU DE L'ALPHABET
110 REM BASIC SIMPLE
120 REM =================
130 REM COULEURS ET FORMES
140 REM ===================
150 DIM A(90)
160 CALL SCREEN(14)
170 CALL COLOR(13,12,14)
180 CALL COLOR(2,2,11)
190 FOR I=5 TO 8
200 CALL COLOR(I,2,11)
210 NEXT I
220 CALL CLEAR
230 CALL CHAR(128,"00000000000000FF")
240 CALL CHAR(129,"FF")
250 CALL CHAR(130,"0101010101010101")
260 CALL CHAR(131,"8080808080808080")
270 CALL HCHAR(6,7,128,21)
280 CALL HCHAR(5,6,128,23)
290 CALL HCHAR(19,7,129,21)
300 CALL HCHAR(20,6,129,23)
310 CALL VCHAR(7,6,130,12)
320 CALL VCHAR(6,5,130,14)
330 CALL VCHAR(7,28,131,12)
340 CALL VCHAR(6,29,131,14)
350 REM ===================
360 REM INITIALISATIONS
370 REM ===================
380 DIM B(27)
```

Suite de la page 33

```
390 FOR I=0 TO 25
400 B(I)=65+I
410 NEXT I
420 B(26)=46
430 FOR X=0 TO 25
440 RANDOMIZE
450 J=INT((27-X)*RND)
460 A(B(J))=X
470 L=INT(X/7)
480 C=X-7*L
490 CALL HCHAR(8+3*L,8+3*C,B(J))
500 FOR K=J TO 26-X
510 B(K)=B(K+1)
520 NEXT K
530 NEXT X
540 A(B(0))=26
550 CALL HCHAR(17,23,B(0))
560 REM ===================
570 REM VERIFICATION DE LA VICTOIRE
580 REM ===================
590 FOR I=65 TO 90
600 IF A(I)<>I-65 THEN 670
610 NEXT I
620 IF A(46)<>26 THEN 670
630 PRINT "BRAVO ! "
640 PRINT
650 GOTO 650
660 REM ===================
670 REM ORDRES
680 REM ===================
690 CALL KEY(5,CHOIX,S)
700 IF S=0 THEN 690
710 CALL KEY(0,ARRET,S)
720 IF S=-1 THEN 710
730 X=A(CHOIX)
740 L=INT(X/7)
750 C=X-7*L
760 LL=L-1
770 CC=C
780 CALL GCHAR(8+3*LL,8+3*CC,Z)
790 IF Z=32 THEN 930
800 LL=L
810 CC=C-1
820 CALL GCHAR(8+3*LL,8+3*CC,Z)
830 IF Z=32 THEN 930
840 LL=L
850 CC=C+1
860 CALL GCHAR(8+3*LL,8+3*CC,Z)
870 IF Z=32 THEN 930
880 LL=L+1
890 CC=C
900 CALL GCHAR(8+3*LL,8+3*CC,Z)
910 IF Z=32 THEN 930
920 GOTO 690
930 A(CHOIX)=7*LL+CC
940 CALL HCHAR(8+3*L,8+3*C,32)
950 CALL HCHAR(8+3*LL,8+3*CC,CHOIX)
960 GOTO 590
```

Le Club TISOFT - HOME se présente

Depuis cinq ans, le club TISOFT existe en Belgique. La fondation du club avait pour raison le manque d'informations et de logiciels pour les calculatrices programmables Texas-Instruments. Le principe de fonctionnement du club repose sur l'échange de programmes entre les membres. Cela implique que nos membres envoient leurs programmes au club et, si ces programmes sont acceptés, ils reçoivent deux programmes au choix en échange.

Nous croyons que ce principe a été couronné d'un vif succès car le club possède en ce moment une bibliothèque de plus de 1000 programmes en tous genres pour les calculatrices programmables.

Dès que l'ordinateur individuel TI-99/4A a fait son apparition sur le marché européen, les mêmes problèmes se sont posés. Le club TISOFT a donc décidé de créer une subdivision TISOFT-HOME, en se basant sur les mêmes principes.

La décision a été prise il y a quelques mois; le catalogue comporte déjà quelques deux cents programmes en tous genres. Ce nombre ne tardera pas à augmenter, vu que le club possède en ce moment plus de 500 membres qui nous envoient constamment des programmes. Comme nous ne voulons pas priver les possesseurs français d'un TI-99/4A de nos services, nous avons décidé de publier cette annonce dans 99 magazine.

Que peuvent attendre nos membres ?

Nos membres :

- peuvent soumettre leurs programmes en échange ;
- peuvent obtenir des listings de programmes ;
- peuvent obtenir de l'information au sujet de nouveaux programmes ;
- reçoivent le catalogue au moment de leur affiliation et les compléments au moment de la parution ;
- peuvent utiliser nos services pour tout problème de programmation ;
- peuvent commander des programmes du catalogue en dehors de l'échange (avec prise en charge des frais) ;
- peuvent commander des programmes sur mesure (demander les conditions).

En plus, nous publions quatre fois par an une "Newsletter" (en anglais seulement) avec des programmes, des astuces et trucs et des informations générales (nouveaux produits, nouvelles concernant le club, etc...)

Acceptation des programmes :

Pour que nous puissions assurer nos membres de la qualité des programmes se trouvant dans notre catalogue, ceux-ci doivent répondre à certains critères. Il est évident que tout programme doit être accompagné d'une documentation complète et claire, et fonctionner impeccablement. Nous nous fions à nos membres en ce qui concerne l'utilisation de techniques de programmation et l'application de formules.

Les textes affichés à l'écran doivent être en français, anglais, néerlandais ou allemand. Le taux d'échange pour la traduction de programmes est de 1 pour 1.

Autres conditions :

- le programme doit être substantiel (non trivial) ;
- le programme doit présenter certains avantages (diminution du temps de travail, du coût, etc...) ;
- le programme ne doit pas être une copie de programme existant (mais peut en être une amélioration substantielle) ;
- le programme doit être contrôlé et testé par le membre qui le soumet.

Nous acceptons des programmes de tous genres (jeux, programmes éducatifs, programmes scientifiques, etc...).

Suite page 38

Introduction au LOGO

Catherine Berdonneau

Le TI-99 est l'un des tout premiers micro-ordinateurs à disposer d'une version de LOGO. Celle-ci est disponible sous forme d'un module de ROM enfichable, et nécessite l'extension mémoire de 32K. Mais peut-être vous posez-vous des questions :

– LOGO c'est quoi ?
– LOGO pourquoi faire ?
– LOGO pourquoi ?

Qu'est-ce que LOGO ?

Cette question était le titre d'une communication de Gérard Bossuet aux journées micro-ordinateurs organisées par l'I.N.R.I.A. en 1979. Trois aspects essentiels de LOGO étaient alors soulignés :

- LOGO est une philosophie de l'éducation. L'approche LOGO est née au milieu des années 60 à Cambridge, près de Boston (Massachusetts), et le terme même de LOGO a été choisi par Wallace Feurzeig en référence au grec "logos" : raisonnement, discours. La première équipe LOGO, qui s'est constituée chez Bolt, Beranek & Newman, s'intéressait à l'apport de l'informatique à l'enseignement, non pour confier à la machine le soin de transmettre à l'élève des savoirs pré-déterminés, mais pour mettre à la disposition de l'étudiant un outil puissant qui lui permette d'aller explorer librement un certain nombre de domaines de connaissance.

- LOGO est un langage de programmation, de la famille de LISP, interactif, procédural et récursif. Il est suffisamment simple pour être accessible à de jeunes enfants (des classes programment en LOGO dès l'âge de cinq-six ans), et en même temps assez sophistiqué pour soutenir l'intérêt des programmeurs chevronnés.

- LOGO est un ensemble de matériels pédagogiques contrôlés par ordinateur, permettant la mise en évidence des processus mentaux mis en jeu par un enfant pour résoudre les problèmes qu'il se pose et auxquels il propose une solution.

Que peut-on faire avec LOGO ?

Pour les plus jeunes, deux "micromondes" sont particulièrement intéressants : celui de la tortue et celui des lutins (sprites en anglais).

La tortue

La tortue est, à l'origine, un petit robot comportant une plate-forme circulaire montée sur roulettes, munie d'un stylo qui peut être soit en position relevée (bouger la tortue ne laisse pas de trace), soit en position baissée (ce qui permet de tracer... sur le sol ou sur de grandes feuilles de papier). Le tout est protégé par une coupole hémisphérique, qui recouvre éventuellement d'autres "organes" : avertisseur sonore, phare, palpeurs... Moins encombrante, la tortue d'écran a la forme d'un petit triangle. Pour la faire apparaître, il suffit d'écrire :

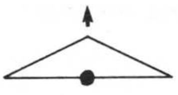
APPELLE TORTUE (ENTER)

Quand on a fini de s'en servir, on s'en débarasse en inscrivant (en un seul mot cette fois) :

PLUS DE TORTUE (ENTER)

Pourquoi un petit triangle ? Pour montrer à la fois où elle est et où elle regarde :

elle regarde par là



elle est ici

Cela fait référence au schéma corporel. Démarrer dans le bon sens est souvent indispensable dans la vie de tous les jours : voyez les problèmes de repérage avec une carte routière, et souvenez-vous de vos ennuis s'il vous est un jour arrivé de prendre une autoroute dans la mauvaise direction !

Avec des ordres très simples (AVANCE, RECULE, DROITE, GAUCHE), nous allons pouvoir faire exécuter à notre tortue des parcours plus ou moins compliqués, et lui faire réaliser des dessins ou tracer des figures géométriques. Après AVANCE ou RECULE, il faut préciser de combien de pas, et après DROITE ou GAUCHE, il faut indiquer de combien de degrés elle doit tourner.

Exemple :

Le point de départ est en bas du mur gauche ; la tortue regarde vers le "haut" de la feuille.

AVANCE 50 DROITE 45
AVANCE 50 DROITE 90
AVANCE 50 DROITE 135
AVANCE 71 RECULE 71
GAUCHE 90 AVANCE 50
DROITE 90 AVANCE 71

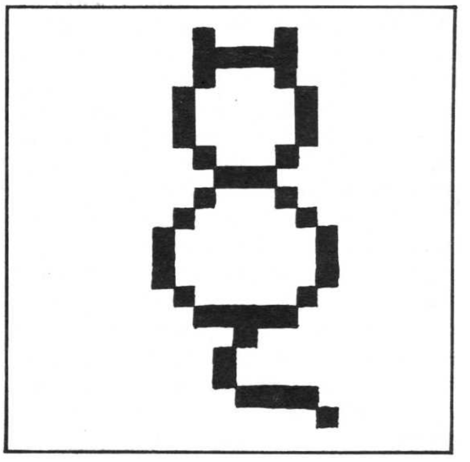
On remarquera que, lorsque la tortue redescend, sa droite et sa gauche sont ma gauche et ma droite, respectivement. Utiliser une tortue amène à beaucoup travailler la latérisation, notion qui n'est souvent pas maîtrisée à la fin de la scolarité primaire : qui de nous n'a pas eu recours au moins une fois au bonhomme d'Ampère ou à la règle des trois doigts pour s'assurer du sens des champs ou du courant dans les problèmes de physique ?

Comme l'ami Pierrot de la chanson, la tortue est munie d'une plume, que l'on peut relever si l'on souhaite un déplacement qui ne laisse pas de trace. Il ne faudra pas oublier de remettre ensuite la plume en position pour écrire !

Le lutin

Un lutin est un être très pudique, qui ne se montre que lorsqu'il est décent. On dispose de 32 lutins, qui peuvent être appelés indépendamment ou à plusieurs à la fois. Il convient d'attribuer à un lutin une forme, choisie parmi 26 possibles (cinq formes sont prédéfinies : un avion, un camion, une fusée, une boule et un carré). Attention ! Le numéro de la forme n'a rien à voir avec le numéro du lutin, de la même façon qu'une voiture R16 peut avoir un numéro d'immatriculation qui ne sera pas "R16", mais plus probablement quelque chose comme 3742 BRD 75. Les 21 autres formes sont à créer par l'utilisateur.

Exemple :



Il faut également donner au lutin une couleur, que l'on choisit parmi les 16 couleurs que gère le TI-99/4A.

Ce n'est qu'après l'avoir ainsi "habillé" qu'on peut appeler le lutin au CENTRE et le rendre visible. On peut alors le déplacer en utilisant les mêmes ordres (AVANCE, RECULE, DROITE, GAUCHE) que pour la tortue. Mais attention ! Un lutin, comme une fée ou un elfe, ne laisse pas de trace de son passage : il n'a pas de plume! Mais on peut aussi lui donner un cap et une vitesse.

On pourra donc réaliser des dessins animés (avec des trucages, qui permettent de transformer un lutin à forme de fleur en un lutin à forme de champignon...), des jeux vidéo, des simulations de phénomènes physiques...

Trois autres "micro-mondes" sont disponibles : les nombres, le texte et la musique.

Les nombres

Les nombres sont des entiers relatifs; on peut les additionner, les soustraire, les multiplier ou les diviser (division euclidienne donnant le résultat entier). On peut aussi les comparer (=,<,>).

Une fonction aléatoire produit à la demande des nombres de un chiffre au hasard entre 0 et 9. Travailler avec des nombres amène donc l'enfant à manipuler régulièrement les faits numériques et logiques dont beaucoup sont encore mal assimilés dans les dernières années de la scolarité des collèges. Par exemple, si l'on veut un nombre au hasard compris entre 0 et 999, il faudra penser à la décomposition décimale d'un nombre à trois chiffres :
C*100+D*10+U
et prendre pour C, D et U ce que fournit la fonction aléatoire.

Le texte

Le texte est traité en LOGO de deux manières : l'objet considéré peut être un mot (chaîne de caractères terminée par un espace) ou une liste (suite d'objets qui peuvent être soit des mots soit des listes). On peut afficher ces objets, en extraire des morceaux, recoller plusieurs morceaux pour constituer un objet plus long. Sensibilisation à la mise en page (présentation), apprentissage des règles de séparation des mots pour donner une bonne lisibilité, organisation et analyse d'un texte, motivation à l'expression écrite, voilà des qualités d'ordre général sur lesquelles vont porter les efforts d'enfants qui explorent ce micro-monde.

La musique

La musique attire l'attention sur les rythmes, et sur leur influence sur notre perception des hauteurs des sons (notes) : il s'agit là d'une approche de l'essentiel du domaine musical, qui n'a rien à voir avec le simple apprentissage du solfège. On peut superposer jusqu'à trois voies sonores et, si l'on joue soi-même d'un instrument (guitare, flûte à bec...), on pourra composer des accompagnements et explorer ainsi le monde de l'harmonie. On peut travailler dans ce micro-monde sans connaître les notes : on exerce son oreille à écouter de manière critique ce que l'on vient de rédiger.

Pourquoi choisir LOGO ?

Il y a trois raisons majeures pour préférer LOGO à d'autres langages :

- LOGO a été conçu pour être utilisé par des enfants, même très jeunes. C'est un langage de programmation sous-tendu par une pédagogie libérale, largement influencée par Piaget.
- LOGO est formateur, non seulement dans les disciplines qui correspondent à ses micro-mondes, mais également d'une manière générale.
- LOGO est une excellente initiation à la programmation structurée.
- LOGO est utilisable par des enfants, même très jeunes.

Aux Etats-Unis, plusieurs écoles et de nombreuses familles utilisent LOGO sur le TI-99 avec des enfants dès l'âge de trois ans. Il ne s'agit pas, en général, d'enfants surdoués. Ces enfants vont d'abord utiliser des programmes écrits en LOGO, qui leur permettront de manipuler lutins ou tortue. D'autres programmes peuvent être conçus facilement (par exemple par un grand frère ou une grande sœur de 10-12 ans) pour associer à une lettre un objet dont le nom commence par cette lettre : A comme AVION, B comme BATEAU, C comme CAMION... Au lieu d'une simple lettre, on utilisera ensuite une série de programmes où le mot doit être tapé en entier : des fiches cartonnées portant ces mêmes mots seront à la disposition de l'enfant qui pourra les recopier à sa guise et associer ainsi le graphisme écriture au graphisme objet.

Dès qu'ils savent lire et écrire, les enfants peuvent se lancer dans des activités passionnantes avec les lutins pour raconter des histoires, les animer et, pourquoi pas, les accompagner d'un fond musical.

Le langage LOGO en lui-même est composé de mots français : pas d'obscurité sur la signification des mots employés, pas de risque d'estropier la prononciation ou la signification courante de mots d'une langue étrangère. Le langage interprété permet l'exécution immédiate de chaque commande tapée au clavier. La notion d'erreur est absente de l'approche LOGO : le terme même d'erreur est exclu de son vocabulaire. Vous ne verrez jamais un message du style de "ERREUR No 20" ou "ERREUR DE SYNTAXE", mais toujours une indication courtoise du type de problème rencontré par l'ordinateur :

- TU N'AS PAS OUVERT LES CROCHETS
- TU ESSAIES DE DIVISER PAR ZERO
- PAS DE FICHIER DISPONIBLE
- VERIFIE LA DISQUETTE
- PAS ASSEZ DE DONNEES
- LA PHRASE EST TROP LONGUE
- "SINON" EST MAL PLACE

Si vous utilisez un mot, par exemple "BIDULE", que la machine ne connaît pas, elle fait preuve de bonne volonté :

APPRENDS-MOI BIDULE

Vous pouvez donc redéfinir les primitifs (mots de base connus par l'ordinateur) qui ne vous plaisent pas : si la plume de la tortue ne vous convient pas, vous lui apprendrez LEVECRAYON et POSECRAYON, ou LEVESTYLO et POSESTYLO au lieu de LEVEPLUME et POSEPLUME.

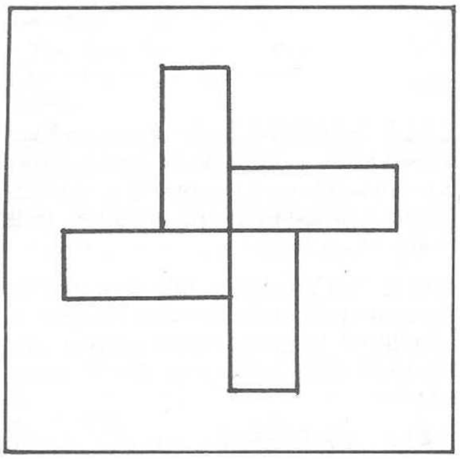
De même, lorsque vous aurez conçu quelque chose, vous pourrez lui apprendre à le réaliser. C'est tout simplement écrire un programme, me direz-vous ! Oui, et non... Je m'explique : essayez donc d'expliquer à un enfant de sept ans ce qu'est un programme; vous risquez fort d'y passer la nuit car, une explication entraînant une autre question, sa curiosité épuisera probablement vos réserves d'imagination et de transposition, sans pour autant avoir été satisfaite.
Si, au contraire, l'enfant dessine un rectangle :

```
AVANCE 20  DROITE 90
AVANCE 35  DROITE 90
AVANCE 20  DROITE 90
AVANCE 35
```

et que là il tape :
RECTANGLE
la réponse :
APPRENDS MOI RECTANGLE
n'engage pas sur le chemin d'un discours mais débouche immédiatement sur l'action :

```
POUR RECTANGLE
AVANCE 20  DROITE 90
AVANCE 35  DROITE 90
AVANCE 20  DROITE 90
AVANCE 35
FIN
```

Désormais, le mot RECTANGLE est reconnu par la machine. Si l'on tape quatre fois ce mot, on obtient une rosace



Ce mot peut à son tour être enseigné à la machine :

POUR ROSACE
REPETE 4 [RECTANGLE]
FIN

Enseigner un mot à la machine consiste donc simplement à lui indiquer dans l'ordre les instructions qu'elle doit exécuter. Pas de numérotation de lignes, pas de virgule, deux points ou autres signes de ponctuation pour séparer les diverses instructions. LOGO est un langage procédural car il fonctionne à partir de la détermination de procédures telles que RECTANGLE, ROSACE, etc...

Autre avantage d'un langage procédural : on peut travailler en même temps sur des problèmes très différents (c'est une aubaine lorsque plusieurs enfants se partagent le même matériel). Aucun besoin de prévoir des sauvegardes continuelles sur disquette ou cassette. Pas de risque de voir un NEW intempestif effacer tout le travail de l'utilisateur précédent. Pas de danger que des numérotations rivales sèment la panique dans les programmes. Chacun utilise les noms dont il a besoin comme titres pour ses propres procédures. Et si d'aventure vous prenez un mot qui sert à un autre usage en ce moment, vous trouvez le texte intégral de sa procédure sous vos yeux : il est alors simple de choisir un autre nom pour sa propre procédure (cela arrive fréquemment par exemple avec CARRE qui, pour l'un, est une figure géométrique et, pour l'autre, une élévation à la puissance deux), ou négocier à l'amiable avec votre concurrent un changement de son titre.

Pas de contrainte autre que celles que l'on se donne volontairement. Pas de seuil, pas de plafond. On apprend en explorant, en tâtonnant, en observant, en manipulant, et non en subissant un enseignement. C'est un savoir librement construit, toujours en évolution, et non un lot de connaissances transmises.

Intérêt de LOGO

LOGO est formateur :

Les principales qualités développées sont ordre et méthode, rigueur, imagination et esprit d'initiative. La mise au point des procédures, le debugging, la conception de nouveaux projets entraînent toujours plus loin.

Par ailleurs, et considérant le seul domaine de la tortue, c'est-à-dire la discipline des mathématiques, cette petite bête à l'aspect inoffensif ouvre des horizons insoupçonnés. En utilisant l'idée même de la tortue, sans plus se laisser freiner par les contingences matérielles d'une tortue d'écran gérée par un micro-ordinateur, on peut explorer des domaines complexes de géométrie ou de physique. Parmi les questions simples, on abordera les déplacements dans le plan : translations, rotations, symétries... Mais si l'on s'affranchit de la contrainte du plan, on utilisera ce véritable outil intellectuel pour des excursions vers la géométrie dans l'espace, la géométrie différentielle, les géométries non euclidiennes... La physique aussi bénéficie de l'apport de la tortue, non seulement la dynamique de Newton, mais même la relativité. Au Massachusetts Institute of Technology, qui compte parmi les plus prestigieuses universités américaines, des cours de niveau maîtrise, en mathématiques et en physique, font appel au concept de la tortue.

Initiation à la programmation structurée

Bien que le but ne soit jamais, avec de jeunes enfants, de leur enseigner explicitement l'informatique, bon nombre de concepts fondamentaux sont abordés au cours des diverses manipulations, pour la plupart avec des enfants ayant moins de 9-10 ans. Tout d'abord une démystification complète de l'ordinateur, qui n'est pas un superman, mais une machine absolument idiote, à laquelle il faut tout indiquer, même ce qui est totalement évident.

Que peut-on découvrir à cet âge comme notions de base ?

- *Procédure* (ou programme). Le fait qu'une procédure peut en appeler une autre : au lieu de se compliquer l'existence à numéroter les lignes et flécher le jeu de piste avec des GOTO, qui rendent rapidement n'importe quel programme peu lisible, chaque tâche est assignée à une procédure, et les enchaînements s'ensuivent logiquement. Toute procédure peut en effet être utilisée au même titre que n'importe quelle instruction, soit en mode direct, soit dans la définition d'une autre procédure.

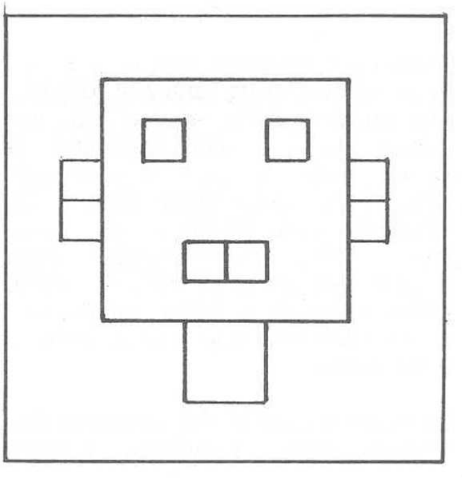
- *Récursion* : une procédure qui s'appelle elle-même. Comme Pascal, LOGO est un langage récursif. Voici un exemple pour le tracé d'une courbe en flocons de neige avec la tortue :

POUR FLOCON :TAILLE :PROFONDEUR
REPETE 3[COTE :TAILLE :PROFONDEUR DROITE 120]
FIN

POUR COTE :LONGUEUR :NIVEAU
SI :NIVEAU = 0 ALORS AVANCE :LONGUEUR STOP
COTE :LONGUEUR/3 :NIVEAU - 1
GAUCHE 60
COTE :LONGUEUR/3 :NIVEAU - 1
DROITE 120
COTE :LONGUEUR/3 :NIVEAU - 1
GAUCHE 60
COTE :LONGUEUR-3 :NIVEAU - 1
FIN

Au niveau 0 de profondeur (qui mesure le nombre d'appels récursifs) le flocon est un simple triangle équilatéral. Au niveau 1, on a un flocon de neige relativement simple.
Au niveau 3, cela commence déjà à être nettement plus compliqué. Pourtant le programme, lui, n'a pas changé. Cette procédure peut être mise au point par un adolescent.

- *Paramètre* : un paramètre est un mot précédé de ":" (dans notre exemple TAILLE, NIVEAU, LONGUEUR et PROFONDEUR). C'est une valeur que l'on déterminera de façon définitive au moment même de l'exécution. Cela permet par exemple d'avoir des carrés plus ou moins grands pour dessiner une tête de robot.



Les noms des paramètres, comme ceux des procédures, peuvent être aussi explicites qu'on le souhaite. Les paramètres locaux sont liés à la procédure dans laquelle ils sont utilisés. C'est ce qui explique que l'on peut changer le nom entre une procédure et une sous-procédure (lors de l'exécution, la valeur assignée à :TAILLE passe à :LONGUEUR, et celle de :PROFONDEUR à :NIVEAU).

- *Conditions* : SI...ALORS...SINON :

Après SI, on doit trouver une expression susceptible d'avoir la valeur VRAI ou FAUX; après ALORS ou SINON, n'importe quelle instruction ou procédure (autres formulations : TESTE, SIVRAI, SIFAUX).

- *Utilisation d'un éditeur* : LOGO dispose d'un éditeur pleine page en temps réel, qui facilite grandement la mise au point des procédures et le debugging. Tout ce qui, dans Pascal, semble compliqué à cause de la nature compilée du langage, est immédiatement manipulable avec LOGO. L'utilisateur peut alors accorder toute son attention à la compréhension du fonctionnement des instructions et à la création de ses propres images mentales pour chaque situation.

Convaincu ? Non ? Mieux qu'un long discours, il vous faut essayer LOGO. Mais attention : LOGO est dangereux ! Si vous avez des enfants, prévoyez vite d'acheter un deuxième ordinateur familial pour avoir encore accès à ce matériel chez vous...

Suite de la page 11

Instructions de BASIC Etendu

ACCEPT AT(x,y):A : le pointeur se trouve au point désigné dans l'écran par les coordonnées x et y. La saisie de la valeur de la variable A est faite à partir de ce point.

VALIDATE(genre, chaîne) : cette instruction s'utilise en association avec la précédente. Elle détermine le type de réponse possible. Le "genre" est soit un chiffre de 0 à 9, soit un caractère numérique (les chiffres et les signes opératoires), soit encore un caractère alphabétique ou alphanumérique.
Si, au lieu d'un genre, on indique une chaîne de caractères entre guillemets, seuls des caractères appartenant à cette chaîne pourront être entrés.

DISPLAY AT(x,y):A : affiche la valeur de A à partir du point de coordonnées x,y.

CALL SOUND(durée,fréquence1,volume1,fréquence2,volume2,fréquence 3,volume3,bruit4,volume4) : fait entendre simultanément jusqu'à trois notes et un bruit.

CALL SAY(mot) : fait dire par le Speech-synthétiser le mot désigné, à condition toutefois que celui-ci soit compris dans le lexique de ce dispositif.

CALL PATTERN : donne à un sprite une forme déterminée par ailleurs.

CALL JOYST(numéro du joyst, A,B) : donne à A et B les valeurs 0,4 ou −4 en fonction de la position du levier du joystick désigné.

CALL COINC : examine s'il y a coïncidence dans la position de deux sprites désignés. Dans ce cas, il donne la valeur −1 à une variable, 0 dans le cas contraire.

Suite de la page 34

Comment devenir membre ?

Le prix d'affiliation est :
- pour la Belgique : 800 FB par an
- pour l'étranger : 1000 FB par an.

Le paiement doit se faire par virement international libellé en Francs Belges, ou par virement à notre compte-chèque postal :
000-1149689-45 TISOFT-HOME
Tennisstraat 16, B-2610 WILRIJK (BELGIQUE)

Adresse pour échange de programmes et correspondance :
TISOFT-HOME c/o Thomas Coppens, boîte postale 63, B-2080 KAPELLEN (BELGIQUE).

Veuillez noter que nous ne pouvons pas accepter les envois recommandés.

Si vous êtes intéressés, n'hésitez pas à devenir membres ou à nous demander de plus amples informations !

Le club d'utilisateurs TI-Education

Vous ne connaissez rien à la micro-informatique, vous sentez le formidable attrait qu'elle exerce sur vos élèves, vous voulez maintenir le contact avec eux tout en vous enrichissant; un vocabulaire étrange (et étranger) vous assaille et vous souhaitez pouvoir le démystifier; vous voulez dominer le BASIC, le Pascal, l'assembleur et le Logo, ou pouvoir simplement bénéficier des travaux de vos collègues.

Deux cents enseignants de toutes disciplines, lettres, technologie, langues, maths, physique, travaux manuels, ayant les mêmes préoccupations, ne connaissant rien à l'informatique en octobre 1982, ont maintenant produit, après une année de fréquentation plus ou moins régulière du club de Vélizy, une centaine de programmes touchant à des domaines variés et, surtout, ont maintenant complètement démystifié la machine...

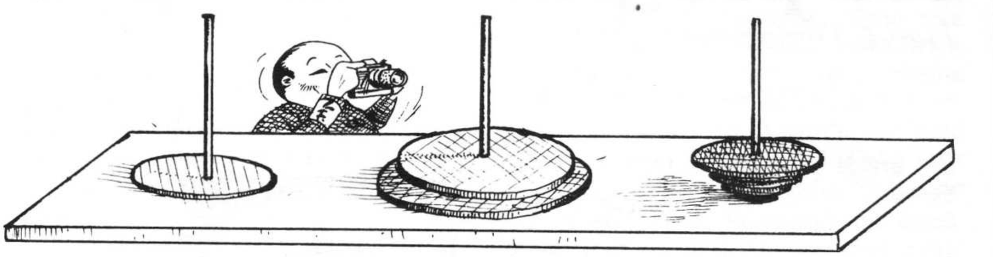
Enseignants de Paris, Strasbourg, Lyon, Rennes, Toulouse et Nice, un nouveau club TI-Education entièrement équipé sera ouvert à tous ceux qui en feront la demande (et gratuitement) dès la rentrée prochaine. Prenez vos précautions et écrivez-nous dès à présent.

Il est évident que les connaisseurs, les mordus et tous ceux qui souhaitent disposer d'un matériel pour réaliser un objectif, en discuter, établir des contacts, entretenir des échanges, etc, seront les bienvenus.

Club TI-Education
Texas-Instruments France
8-10 avenue Morane-Saulnier
78140 VELIZY

Les tours de Hanoï

Denise Amrouche



Vous connaissez probablement ce petit casse-tête se présentant généralement sous la forme d'un disque de bois dans lequel sont plantées trois tiges cylindriques disposées en triangle. Sur l'une d'entre elles sont enfichés cinq ou six anneaux de taille décroissante, le plus petit étant sur le dessus.

Le jeu consiste, en prenant à chaque coup un seul anneau et en le mettant dans une colonne vide ou sur un anneau plus grand, à reconstituer la pyramide complète sur une autre tige.

Dans la version pour TI-99 présentée ici, les anneaux sont représentés par des barres de couleurs et de dimensions différentes. Les trois tiges, au lieu d'être représentées en triangle, sont disposées en ligne, ce qui simplifie sensiblement la représentation graphique du jeu.

Le joueur choisit de jouer avec 4, 5 ou 6 anneaux au départ, ce qui permet de varier le niveau de difficulté. Initialement, tous les anneaux se trouvent sur la pile 1, le jeu consistant à les disposer dans le même ordre sur l'une des deux autres piles. Pour jouer chaque coup, le joueur indique en premier le numéro de la pile de départ du coup à jouer, puis celui de la pile d'arrivée. Tous les coups sont permis, sauf de poser un anneau sur un anneau plus petit. A la fin du jeu le score est indiqué, c'est-à-dire le nombre de coups qu'il a fallu au joueur pour transférer la pyramide d'une pile dans une autre.

Dans le programme, les anneaux sont numérotés de 1 à 6, l'anneau de numéro I ayant un diamètre de $(2\star I-1)$. Pour chacun des anneaux, la ligne 420 définit le code ASCII du caractère CAR(I) qui le représentera, le groupe de couleurs GRP(I) correspondant et la couleur COUL(I) choisie.

Dans le but de mémoriser, à chaque étape, le contenu d'une pile, on juxtapose les numéros des anneaux constituant cette pile de haut en bas. Par exemple, si la pile 1 contient de haut en bas les anneaux 2, 3 et 5, on aura PILE(1)=2.35. Si l'anneau 2 est enlevé, il suffira de faire $(\text{PILE}(1)-2)\star 10$ pour obtenir la nouvelle valeur de PILE(1), soit 3.5 (ligne 860). Si l'anneau 1 est amené sur cette pile, la nouvelle valeur de PILE(1) sera $\text{PILE}(1)\star 0.1+1$, soit 1.35 (ligne 830).

Un autre problème se présente pour séparer clairement les deux introductions de données par le clavier : pile de départ et pile d'arrivée. Sans certaines précautions, si le doigt reste appuyé sur la touche assez longtemps, le TI-99 enregistrera la même donnée pour la pile de départ et la pile d'arrivée. La ligne 590 enregistre la pile de départ, la ligne 630 est exécutée tant que la pression sur la touche n'a pas été relâchée. Il en va de même pour les lignes 700 à 730.

Le compteur C enregistre le nombre de coups licites effectués, dont la valeur finale sera affichée en fin de partie. Seuls les coups licites sont comptés : les coups illicites, n'étant pas exécutés, ne sont pas enregistrés par ce compteur.

Ce jeu est, au départ, un véritable casse-tête et il n'est pas rare de voir un joueur inexpérimenté, devant une pile de 6 anneaux, dépasser la centaine de coups sans pour autant parvenir au but. Mais il y a un truc, simple et purement mathématique, qui permet de résoudre ce jeu avec aisance.

Le programme que je vous ai présenté ici peut être modifié. Vous pouvez changer à votre goût la représentation graphique; vous pouvez aussi adapter le jeu pour qu'il fonctionne avec 7 anneaux ou plus (le principe de résolution du jeu reste strictement le même quel que soit le nombre d'anneaux utilisés).

Alors, amusez-vous bien!

```
50 REM =====================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET DENISE AMROUCHE
80 REM =====================
90 REM
100 REM ********************
110 REM
120 REM   TOUR DE HANOI
130 REM
140 REM ********************
150 REM BASIC SIMPLE
160 CALL SCREEN(16)
170 CALL CLEAR
180 PRINT "NOMBRE D'ANNEAUX?(4,5,6)"
190 CALL KEY(0,K,S)
200 IF S=0 THEN 190
210 CALL SOUND(500,262,1,330,1,392,1)
220 IF K<52 THEN 190
230 IF K>54 THEN 190
240 N=K-48
250 IF (N-4)*(N-6)>0 THEN 180
260 REM ****INITIALISATIONS*
270 IF N>4 THEN 300
280 CODE=1.234
290 GOTO 340
300 IF N>5 THEN 330
310 CODE=1.2345
320 GOTO 340
330 CODE=1.23456
340 PILE(1)=CODE
350 PILE(2)=0
360 PILE(3)=0
370 HAUT(1)=10
380 HAUT(2)=10+N
390 HAUT(3)=10+N
400 C=0
410 DATA 136,14,10,128,13,5,120,12,12,11
2,11,4,104,10,14,96,9,13
420 FOR I=1 TO 6
430 READ CAR(I),GRP(I),COUL(I)
440 CALL COLOR(GRP(I),COUL(I),COUL(I))
450 NEXT I
460 RESTORE
470 CALL CLEAR
480 FOR I=1 TO N
490 CALL HCHAR(9+I,8-I,CAR(I),2*I-1)
500 NEXT I
510 CALL HCHAR(17,7,49)
520 CALL HCHAR(17,17,50)
530 CALL HCHAR(17,27,51)
540 REM ******** JEU *******
550 CALL HCHAR(20,9,68)
560 CALL HCHAR(20,10,69)
570 CALL HCHAR(20,11,63)
580 CALL KEY(3,DE,STATUS)
590 IF STATUS=0 THEN 580
600 REM VERIFIE QUE LA TOUCHE EST RELACH
EE
610 CALL KEY(3,ARRET,STATUS)
620 IF STATUS=-1 THEN 610
630 CALL HCHAR(20,13,DE)
640 DE=DE-48
650 CALL SOUND(100,110,3)
660 IF (DE<1)+(DE>3)+(PILE(DE)=0)THEN 89
0
670 CALL HCHAR(20,16,65)
680 CALL HCHAR(20,17,63)
690 CALL KEY(3,A,STATUS)
700 IF STATUS=0 THEN 690
710 CALL KEY(3,ARRET,STATUS)
720 IF STATUS=-1 THEN 710
730 CALL HCHAR(20,19,A)
740 A=A-48
750 CALL SOUND(100,262,2)
760 IF (A<1)+(A>3)THEN 890
770 IF PILE(A)=0 THEN 790
780 IF (PILE(DE)>PILE(A))THEN 890
790 C=C+1
800 U=INT(PILE(DE))
810 HAUT(A)=HAUT(A)-1
820 PILE(A)=.1*PILE(A)+U
830 CALL HCHAR(HAUT(DE),10*DE-2-U,32,2*U
-1)
840 HAUT(DE)=HAUT(DE)+1
850 PILE(DE)=10*(PILE(DE)-U)
860 CALL HCHAR(HAUT(A),10*A-2-U,CAR(U),2
*U-1)
870 IF PILE(2)=CODE THEN 910
880 IF PILE(3)=CODE THEN 910
890 CALL HCHAR(20,7,32,20)
900 GOTO 550
910 PRINT "BRAVO!!                  V
OUS AVEZ GAGNE EN          ";C;"COUPS"
920 PRINT
930 PRINT
940 PRINT
950 PRINT
960 PRINT "UNE AUTRE PARTIE?(O/N)"
970 CALL KEY(0,D,E)
980 IF E=0 THEN 970
990 IF D=79 THEN 120
```

Récupérez les programmes Apple

Hervé Thiriez

Cet article s'adresse à tous ceux qui désirent adapter des programmes écrits en Applesoft au TI Basic. En effet, aucun ordinateur au monde ne possède à ce jour autant de programmes que l'Apple : autant ne pas s'en priver et savoir adapter les programmes écrits pour Apple afin qu'ils marchent sur notre 99/4A préféré. Nous allons voir, par conséquent, comment effectuer la conversion du Basic Aplesoft vers le TI Basic.

Instrutions similaires

Les instructions qui suivent appartiennent aux deux Basics et y jouent un rôle soit identique, soit fort voisin. Nous commenterons par la suite les principales différences. Nous ne pouvons prétendre à l'exhaustivité : il faudrait y consacrer une centaine de pages ...

ABS, ASC, ATN, CHR$, COS, DATA, DEF, DIM, END, EXP, FOR .. TO, GOSUB, GOTO, On .. GOTO, ON .. GOSUB, READ, REM, RETURN, SGN, SIN, SQR, STEP, STOP, STR$ et TAN.

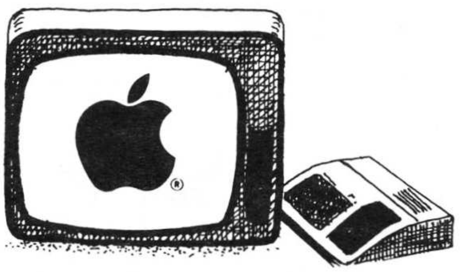
Instructions différentes

La table ci-dessous indique la correspondance entre les deux langages pour les instructions divergeant sensiblement l'une de l'autre.

Basic Applesoft TI Basic

Touche RESET	Touche CLEAR
Touche RETURN	Touche ENTER
HOME	CALL CLEAR
INPUT "texte";A	INPUT "texte" :A
PRINT CHR$(4)	"CLOSE" :NEW BYE
STOP	BREAK (différent)
NOTRACE	UNTRACE
IF ... THEN ...	IF .. THEN .. ELSE
MID$	SEG$
VTAB 6	PRINT : : : : : :

Nous commentons par la suite les principales différences d'utilisation entre les Basics de l'Apple et du TI-99.



Instructions multiples

Applesoft, comme le Basic Etendu, mais contrairement au TI Basic, autorise les instructions multiples sur une ligne. Ces instructions sont séparées par des ":". La possibilité d'avoir des instructions multiples compense l'inexistence du IF .. THEN .. ELSE, car on peut écrire par exemple :

```
10 IF A=3 THEN B=A+2 :C=5 :
    GOTO 30
20 B= 1
```

là où il faudrait en TI Basic :

```
10 IF A= 3 THEN 15 ELSE 20
15 B= A+ 2
16 C= 5
17 GOTO 30
20 B= 1
```

Respect de l'espacement

Le BASIC Applesoft est tokenisé, c'est-à-dire que chaque mot réservé de l'Applesoft est recodé de façon interne en un caractère spécial. Il est par conséquent possible d'accoler dans une instruction des noms de variables : l'interpréteur redécoupe le programme en isolant les mots réservés du Basic. On peut ainsi écrire "10 PRINTA"; si l'on liste cette instruction, on obtiendra "10 PRINT A". En TI Basic, les séparations par des blancs ne peuvent être omises, et "10 PRINTA" sera considéré comme une erreur de syntaxe.

Faire bonne impression...

L'ordre PRINT peut s'écrire de façon abrégée avec le signe "?" sur Apple. Cela permet d'utiliser l'ordinateur comme calculatrice sans perdre de temps avec l'entrée complète de l'ordre PRINT. C'est malheureusement impossible en TI Basic.

Il n'est pas obligatoire en Applesoft de séparer différents éléments devant être imprimés à la queue leu leu par le signe ";". Ainsi, "PRINT A$;B$" ou "PRINT AB" sont équivalents pour l'Applesoft. Le second mode d'écriture est interdit en TI Basic, pour lequel les séparateurs ";" sont obligatoires.

La syntaxe de l'ordre PRINT diffère aussi d'un Basic à l'autre. Ainsi, "PRINT 3 :4" imprime-t-il 3 et 4, sur deux lignes successives, en TI Basic. Cela donnerait une erreur de syntaxe à son concurrent qui, pour imprimer de la même manière un "3" et un "4", a le choix entre "PRINT 3 : PRINT 4" et "PRINT 3 CHR$(13) 4".

Impression de valeurs

Lors de l'impression de valeurs en TI Basic, chaque valeur numérique est encadrée par un blanc à gauche et un blanc à droite. Ainsi, en TI Basic :
PRINT "A=";23;"!" donnera "A=23 !"
En Applesoft, on obtiendrait "A=23!"
Pour obtenir le même résultat en TI Basic, il faudrait écrire :
PRINT "A=";STR$(23);"!"

Particularités des boucles

Il n'est pas nécessaire en Applesoft de spécifier l'indice d'une boucle au niveau d'un NEXT. Ainsi "FOR I=1 TO 10 ... NEXT" est-il équivalent à "FOR I=1 TO 10 ... NEXT I". Il n'y a en effet aucune ambiguité, dans la mesure où les boucles doivent forcément être imbriquées. Le TI Basic exige la spécification de l'indice dans le NEXT.

En TI Basic, une boucle "FOR I=2 TO 1" ne s'exécute pas, et l'on saute immédiatement à la suite du "NEXT I". Par contre, Applesoft passe une fois dans la boucle quand l'indice initial est supérieur à l'indice final.

Interruption

L'interruption en Applesoft s'opère avec CTRL-C (maintenir la touche

CTRL enfoncée, puis appuyer sur C), le programme pouvant alors être relancé en tapant "CONT", sauf si l'interruption était survenue dans une boucle. Pour le TI Basic, l'interruption est provoquée par la touche CLEAR, la relance du programme étant obtenue en tapant "CONTINUE". Par contre, la relance fonctionne même si l'interruption a eu lieu pendant une boucle.

Gestion de l'écran

La présentation à l'écran, lors de l'exécution d'un programme en Applesoft, est différente : le curseur se place initialement en haut et à gauche de l'écran, et non en bas comme avec le TI-99. Si vous voulez comprendre et transposer les programmes Apple, il faut connaître certaines des instructions de gestion d'écran parmi les plus fréquentes :

HOME	: CALL CLEAR, curseur en haut
FLASH	: écrit à l'écran en clignotant
INVERSE	: écrit en noir sur fond blanc
NORMAL	: retour à la normale
POKE 32,G	: marge de gauche à G
POKE 33,D	: marge de droite à D
POKE 34,H	: marge du haut à H
POKE 35,B	: marge du bas à B
TEXT	: annule les POKEs précédents
CALL -936	: HOME (autre manière)
CALL -868	: blanchit le reste de la ligne
CALL -912	: remonte l'écran d'une ligne
POS(0)	: position du curseur (0 à 39)
PEEK(-16336)	: "click" sonore

A l'aide des POKEs, il est possible de définir une fenêtre dans laquelle défile le texte à l'écran. Tout ce qui se trouve en dehors de la fenêtre reste tel quel jusqu'au moment où de nouveaux POKEs, ou TEXT, libèrent l'écran.

Avec l'Apple, lorsqu'au cours d'une instruction que l'on modifie à l'écran, on appuie sur la touche RETURN, on garde ce qu'il y avait avant le curseur; le reste est effacé. Par contre, quand on appuie sur ENTER dans la même situation avec le TI-99, le reste de la ligne n'est pas coupé, et ce qui suit le curseur reste valable.

Numéros de ligne et listage

Les numéros de ligne vont de 0 à 63999 en Applesoft, et de 1 à 32767 en TI Basic.

Le TI Basic possède l'instruction NUM (ou NUMBER) pour la numérotation automatique de lignes, ainsi que RES (ou RESEQUENCE) pour la re-numérotation. Ces instructions n'appartiennent pas au Basic Applesoft.

Pour obtenir la liste entre les instructions 20 et 60, il faut faire "LIST 20,60" ou "LIST 20-60" en Applesoft et seulement "LIST 20-60" en TI Basic. Pour ce dernier, quand la limite supérieure (resp. inférieure) de la liste est inférieure (resp. supérieure) au numéro de la première (resp. dernière) ligne du programme, celle-ci est listée. En Applesoft, rien n'est listé dans ce cas.

Précision numérique

Le nombre de chiffres significatifs est différent entre les deux Basics. Ainsi, l'ordre "PRINT 1/7" donne-t-il ".142857143" comme résultat avec l'Apple et ".1428571429" avec le TI-99.

Un nombre réel est limité à 1E38 en Applesoft (10 puissance 38), et à quasiment 1E128 en TI Basic.

Vrai ou faux ?

Dans les deux langages, "PRINT 1=2" donnera le résultat "0", correspondant à la notion de FAUX. On observe par contre des notions différentes du VRAI, qui est codé "+1" en Applesoft et "−1" en TI Basic.

Il faut s'en méfier lors de la transposition, car de nombreux programmes astucieux utilisent les opérations logiques pour faire l'économie de l'instruction IF. Ainsi "A=1+(B=C)" revient à programmer "A=1", puis "IF B=C THEN A=2" en Applesoft. Il suffirait de remplacer le "+" par un "−" pour obtenir un résultat identique en TI Basic.

En matière de logique, il faut faire correspondre au "AND" et au "OR" de l'Applesoft le "*" et le "+" du TI Basic. Ainsi, "IF A=1 AND B=2 THEN ..." se traduit "IF (A=1)*(B=2) THEN ...".

Divergences sur ON X

L'instruction "ON X GOTO ..." engendre un message d'erreur "bad value in X" en TI Basic si la valeur de X est inférieure à 1 ou trop grande. C'est accepté sans contrôle par Applesoft qui saute alors à l'instruction suivante.

Gestion de chaînes

A la place de l'instruction de maniement de chaîne SEG$ du TI Basic, Applesoft propose trois instructions, dont MID$ qui joue le même rôle que SEG$. LEFT$(R$,L) renvoie les L caractères à gauche de la chaîne R$. A l'inverse, RIGHT$(R$,R) fournit les R caractères à droite. On peut donc remplacer LEFT$(R$,L) par SEG$(R$,1,L) et RIGHT$(R$,R) par SEG$(R$,LEN(R$) + 1,L).

L'instruction VAL fonctionne de la même manière tant que son argument contient uniquement une valeur numérique sous forme de chaîne. Pour Applesoft, VAL("25F50") donnera "25" et VAL("ALFRED") vaudra "0" : le processus s'arrête dès qu'un caractère non numérique est rencontré. En TI Basic, l'exécution du programme serait arrêtée dans ces deux cas.

RESTORE ou RESTORE 100 ?

Le RESTORE du TI Basic est plus puissant que celui de l'Applesoft, car il peut débuter à tel ou tel endroit du programme. Le RESTORE de l'Applesoft relance obligatoirement la lecture des DATA à partir du haut du programme.

Graphiques et sons

Les différences au niveau des instructions graphiques et sonores sont tellement importantes que nous ne pouvons pas conseiller de transposition simple. Il vaudrait mieux réécrire totalement les parties de programmes concernées.



Le manque de place nous a contraint à différer la publication de deux articles et à la reporter au numéro 2 de "99". Il s'agit d'un éloge de la touche REDO, par Robert Peltier, et d'une étude de Jean Sahal sur les procédés économiques de sauvegarde de fichiers et programmes.

Nous prions les auteurs de nous excuser.

Le troisième menuet de Bach



Alexandre Duback

Le TI-99 sait faire de la musique. Pour le prouver, voici un petit programme jouant, sur deux voies sonores, le 3e menuet de Jean-Sebastien BACH. C'est d'ailleurs plus facile que d'exécuter le 2e menuet sur trois voies...

Ce programme est écrit en Basic simple. Il ne nécessite donc aucune configuration particulière pour fonctionner.

Les lignes 140 à 190 affichent au centre de l'écran le titre du morceau. Cet affichage, comme dans la plupart des programmes musicaux que nous publions, est réduit au strict minimum. Il ne tient qu'à vous de l'améliorer...

Les lignes 200 à 580 donnent à différentes variables les valeurs de fréquences des notes utilisées tout au long du morceau de musique. Les lignes 600 à 620, quant à elles, donnent les valeurs des différentes durées de notes utilisées. Il est d'ailleurs très significatif du style de Bach, très syncopé, qu'il n'y ait que trois valeurs différentes à ce niveau. Le volume, enfin, déterminé à la ligne 640, consiste en une seule valeur. Cela aussi est assez caractéristique du style du compositeur.

La série des instructions "musicales" débute à la ligne 660 et consiste en toute une suite de CALL SOUND. Profitons-en pour rappeler que les valeurs qui suivent cette instruction se décomposent comme suit :

– durée
– fréquence de la première voie
– volume de la première voie
– fréquence de la deuxième voie
– volume de la deuxième voie
– fréquence de la troisième voie
– volume de la troisième voie

Ainsi, la programmation du son sur une seule voie requiert seulement trois valeurs en argument du CALL SOUND. Vous pouvez vérifier que ce menuet de Bach est bien joué sur deux voies.



Il aurait été possible de concevoir ce programme différemment. Sans prétention artistique aucune, j'ai écrit le petit programme ci-joint qui joue sur une seule voie le thème bien connu de Frère Jacques. Mon propos n'était certes pas d'éblouir par mon talent musical, mais d'illustrer une autre forme de programmation : toutes les données numériques sont regroupées dans des lignes de DATA, puis lues une à une et interprétées avec une seule instruction CALL SOUND, utilisée autant de fois qu'il y a de notes dans le morceau.

Remarquez que, pour me simplifier le travail, j'ai utilisé une seule durée de note, ce qui, sur les notes longues, n'est pas du meilleur effet. Ce programme de Frère Jacques est donc très perfectible :
- on pourrait en premier lieu faire varier la durée de façon plus efficace; il ne serait pas nécessaire d'indiquer la durée à chaque note, mais seulement les changements de durée ;
- on pourrait surtout jouer cet air sur plusieurs voies en utilisant la technique du canon : la première voie joue normalement le thème principal que la seconde voie reprend ensuite avec une mesure de retard,...

Voici présentées deux approches de programmation différentes pour un même résultat. Il en existe sans doute d'autres. Vous avez même peut-être déjà votre idée sur la meilleure façon de programmer un morceau de musique. Envoyez-nous vos créations !



```
50 REM ====================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET L'AUTEUR
80 REM ====================
90 REM
100 REM 3EME MENUET DE BACH
110 REM ==AUTEUR:
120 REM ==TOM NICKEL 1980
130 REM ==
140 CALL CLEAR
150 PRINT "****************************"
160 PRINT "*     3EME MENUET DE BACH  *"
170 PRINT "****************************"
: : : : : : : : :
180 FOR X=1 TO 100
190 NEXT X
200 REM ***** NOTES *******
210 A1=110
220 AS1=117
230 B1=123
240 C1=131
250 CS1=139
260 D1=147
270 DS1=156
280 E1=165
290 F1=175
300 FS1=185
310 G1=196
320 A2=220
330 B2=247
340 C2=262
350 CS2=277
360 D2=294
370 DS2=311
380 E2=330
390 F2=349
400 FS2=370
410 G2=392
420 A3=440
430 B3=494
440 C3=523
450 CS3=554
460 D3=587
470 DS3=622
480 E3=659
490 F3=698
500 FS3=740
510 G3=784
520 A4=880
530 B4=988
540 C4=1047
550 D4=1175
560 E4=1319
570 F4=1397
580 G4=1568
```

```
590 REM **** MESURES *******
600 X=300
610 Y=600
620 Z=1199
630 REM *** VOLUME ***
640 V=0
650 REM ******************
660 FOR A=1 TO 2
670 CALL SOUND(X,G2,V,G1,V)
680 CALL SOUND(X,B3,V,G1,V)
690 CALL SOUND(X,D3,V,G1,V)
700 CALL SOUND(X,G3,V,G1,V)
710 CALL SOUND(X,A3,V,D1,V)
720 CALL SOUND(X,FS3,V,D1,V)
730 CALL SOUND(X,G3,V,G1,V)
740 CALL SOUND(X,G3,V,B1,V)
750 CALL SOUND(X,G2,V,D1,V)
760 CALL SOUND(X,G2,V,G1,V)
770 CALL SOUND(X,G2,V,D1,V)
780 CALL SOUND(X,G2,V,B1,V)
790 CALL SOUND(X,G2,V,G1,V)
800 CALL SOUND(X,B3,V,G1,V)
810 CALL SOUND(X,D3,V,G1,V)
820 CALL SOUND(X,G3,V,G1,V)
830 CALL SOUND(X,A3,V,D1,V)
840 CALL SOUND(X,FS3,V,D1,V)
850 CALL SOUND(X,G3,V,G1,V)
860 CALL SOUND(X,G3,V,B1,V)
870 CALL SOUND(X,G2,V,D1,V)
880 CALL SOUND(X,G2,V,G1,V)
890 CALL SOUND(X,G2,V,D1,V)
900 CALL SOUND(X,G2,V,B1,V)
910 CALL SOUND(Y,E3,V,C1,V)
920 CALL SOUND(Y,E3,V,G1,V)
930 CALL SOUND(X,E3,V,C1,V)
940 CALL SOUND(X,G3,V,C1,V)
950 CALL SOUND(Y,D3,V,B1,V)
960 CALL SOUND(Y,D3,V,G1,V)
970 CALL SOUND(X,D3,V,B1,V)
980 CALL SOUND(X,G3,V,B1,V)
990 CALL SOUND(Y,C3,V,A1,V)
1000 CALL SOUND(X,D3,V,FS1,V)
1010 CALL SOUND(X,C3,V,FS1,V)
1020 CALL SOUND(X,B3,V,G1,V)
1030 CALL SOUND(X,C3,V,G1,V)
1040 CALL SOUND(X,A3,V,D1,V)
1050 CALL SOUND(X,A3,V,E1,V)
1060 CALL SOUND(X,A3,V,FS1,V)
1070 CALL SOUND(X,A3,V,D1,V)
1080 CALL SOUND(X,A3,V,E1,V)
1090 CALL SOUND(X,A3,V,FS1,V)
1100 CALL SOUND(X,G2,V,G1,V)
1110 CALL SOUND(X,B3,V,G1,V)
1120 CALL SOUND(X,D3,V,G1,V)
1130 CALL SOUND(X,G3,V,G1,V)
1140 CALL SOUND(X,A3,V,D1,V)
1150 CALL SOUND(X,FS3,V,D1,V)
1160 CALL SOUND(X,G3,V,G1,V)
1170 CALL SOUND(X,G3,V,B1,V)
1180 CALL SOUND(X,G2,V,D1,V)
1190 CALL SOUND(X,G2,V,G1,V)
1200 CALL SOUND(X,G2,V,D1,V)
1210 CALL SOUND(X,G2,V,B1,V)
1220 CALL SOUND(X,G2,V,G1,V)
1230 CALL SOUND(X,B3,V,G1,V)
1240 CALL SOUND(X,D3,V,G1,V)
1250 CALL SOUND(X,G3,V,G1,V)
1260 CALL SOUND(X,A3,V,D1,V)
1270 CALL SOUND(X,FS3,V,D1,V)
1280 CALL SOUND(X,G3,V,G1,V)
1290 CALL SOUND(X,G3,V,B1,V)
1300 CALL SOUND(X,G2,V,D1,V)
1310 CALL SOUND(X,G2,V,G1,V)
1320 CALL SOUND(X,G2,V,D1,V)
1330 CALL SOUND(X,G2,V,B1,V)
1340 CALL SOUND(Y,E3,V,C1,V)
1350 CALL SOUND(X,D3,V,E1,V)
1360 CALL SOUND(X,C3,V,E1,V)
1370 CALL SOUND(X,B3,V,G1,V)
1380 CALL SOUND(X,A3,V,G1,V)
1390 CALL SOUND(Y,D3,V,B1,V)
1400 CALL SOUND(X,C3,V,D1,V)
1410 CALL SOUND(X,B3,V,D1,V)
1420 CALL SOUND(X,A3,V,G1,V)
1430 CALL SOUND(X,G2,V,G1,V)
1440 CALL SOUND(X,A3,V,C1,V)
1450 CALL SOUND(X,B3,V,C1,V)
1460 CALL SOUND(X,C3,V,C1,V)
1470 CALL SOUND(Y,D2,V,C1,V)
1480 CALL SOUND(Y,FS2,V,D1,V)
1490 CALL SOUND(Y,G2,V,G1,V)
1500 CALL SOUND(Y,G2,V,D1,V)
1510 CALL SOUND(Y,G2,V,G1,V)
1520 CALL SOUND(X,G2,V,E1,V)
1530 CALL SOUND(X,A3,V,E1,V)
1540 CALL SOUND(X,B3,V,DS1,V)
1550 CALL SOUND(X,A3,V,DS1,V)
1560 CALL SOUND(X,G2,V,B1,V)
1570 CALL SOUND(X,FS2,V,B1,V)
1580 CALL SOUND(Y,G2,V,E1,V)
1590 CALL SOUND(Y,E2,V,B1,V)
1600 CALL SOUND(Y,E2,V,E1,V)
1610 CALL SOUND(X,G3,V,E1,V)
1620 CALL SOUND(X,FS3,V,E1,V)
1630 CALL SOUND(X,E3,V,G1,V)
1640 CALL SOUND(X,G3,V,G1,V)
1650 CALL SOUND(X,FS3,V,B2,V)
1660 CALL SOUND(X,E3,V,B2,V)
1670 CALL SOUND(X,FS3.V,B1,V)
1680 CALL SOUND(X,FS3,V,DS1,V)
1690 CALL SOUND(X,B3,V,FS1,V)
1700 CALL SOUND(X,B3,V,B2,V)
1710 CALL SOUND(X,B3,V,FS1,V)
1720 CALL SOUND(X,B3,V,DS1,V)
1730 CALL SOUND(X,G3,V,E1,V)
1740 CALL SOUND(X,FS3,V,E1,V)
1750 CALL SOUND(X,E3,V,G1,V)
1760 CALL SOUND(X,G3,V,G1,V)
```

```
1770 CALL SOUND(X,FS3,V,B2,V)
1780 CALL SOUND(X,E3,V,B2,V)
1790 CALL SOUND(Y,FS3,V,B1,V)
1800 CALL SOUND(Y,B3,V,A2,V)
1810 CALL SOUND(Y,E3,V,G1,V)
1820 CALL SOUND(X,FS3,V,A2,V)
1830 CALL SOUND(X,G3,V,A2,V)
1840 CALL SOUND(X,A4,V,A2,V)
1850 CALL SOUND(Y,B3,V,B2,V)
1860 CALL SOUND(Y,DS3,V,B1,V)
1870 CALL SOUND(Y,E3,V,E1,V)
1880 CALL SOUND(X,DS3,V,E1,V)
1890 CALL SOUND(X,E3,V,E1,V)
1900 CALL SOUND(Y,FS3,V,E1,V)
1910 CALL SOUND(Y,G3,V,B1,V)
1920 CALL SOUND(X,G3,V,D1,V)
1930 CALL SOUND(X,FS3,V,D1,V)
1940 CALL SOUND(X,E3,V,G1,V)
1950 CALL SOUND(X,D3,V,G1,V)
1960 CALL SOUND(Y,E3,V,C1,V)
1970 CALL SOUND(X,E3,V,D1,V)
1980 CALL SOUND(X,D3,V,D1,V)
1990 CALL SOUND(X,C3,V,E1,V)
2000 CALL SOUND(X,B3,V,E1,V)
2010 CALL SOUND(Y,C3,V,A1,V)
2020 CALL SOUND(X,C3,V,B1,V)
2030 CALL SOUND(X,B3,V,B1,V)
2040 CALL SOUND(X,A3,V,C1,V)
2050 CALL SOUND(X,G2,V,C1,V)
2060 CALL SOUND(Y,FS2,V,D1,V)
2070 CALL SOUND(X,E2,V,A1,V)
2080 CALL SOUND(X,FS2,V,A1,V)
2090 CALL SOUND(Y,D2,V,D1,V)
2100 CALL SOUND(X,A3,V,FS1,V)
2110 CALL SOUND(X,A3,V,D1,V)
2120 CALL SOUND(X,D2,V,FS1,V)
2130 CALL SOUND(X,D2,V,D1,V)
2140 CALL SOUND(X,D2,V,FS1,V)
2150 CALL SOUND(X,D2,V,D1,V)
2160 CALL SOUND(X,B3,V,G1,V)
2170 CALL SOUND(X,B3,V,D1,V)
2180 CALL SOUND(X,D2,V,G1,V)
2190 CALL SOUND(X,D2,V,D1,V)
2200 CALL SOUND(X,D2,V,G1,V)
2210 CALL SOUND(X,D2,V,D1,V)
2220 CALL SOUND(Y,C3,V,FS1,V)
2230 CALL SOUND(X,D3,V,D1,V)
2240 CALL SOUND(X,C3,V,D1,V)
2250 CALL SOUND(X,B3,V,G1,V)
2260 CALL SOUND(X,C3,V,G1,V)
2270 CALL SOUND(X,A3,V,D1,V)
2280 CALL SOUND(X,A3,V,E1,V)
2290 CALL SOUND(X,A3,V,FS1,V)
2300 CALL SOUND(X,A3,V,D1,V)
2310 CALL SOUND(X,A3,V,E1,V)
2320 CALL SOUND(X,A3,V,FS1,V)
2330 CALL SOUND(X,G2,V,G1,V)
2340 CALL SOUND(X,B3,V,G1,V)
2350 CALL SOUND(X,D3,V,G1,V)
2360 CALL SOUND(X,G3,V,G1,V)
2370 CALL SOUND(X,A3,V,D1,V)
2380 CALL SOUND(X,FS3,V,D1,V)
2390 CALL SOUND(X,G3,V,G1,V)
2400 CALL SOUND(X,G3,V,B1,V)
2410 CALL SOUND(X,G2,V,D1,V)
2420 CALL SOUND(X,G2,V,G1,V)
2430 CALL SOUND(X,G2,V,D1,V)
2440 CALL SOUND(X,G2,V,B1,V)
2450 CALL SOUND(X,G2,V,G1,V)
2460 CALL SOUND(X,B3,V,G1,V)
2470 CALL SOUND(X,D3,V,G1,V)
2480 CALL SOUND(X,G3,V,G1,V)
2490 CALL SOUND(X,A3,V,D1,V)
2500 CALL SOUND(X,FS3,V,D1,V)
2510 CALL SOUND(X,G3,V,G1,V)
2520 CALL SOUND(X,G3,V,B1,V)
2530 CALL SOUND(X,G2,V,D1,V)
2540 CALL SOUND(X,G2,V,G1,V)
2550 CALL SOUND(X,G2,V,D1,V)
2560 CALL SOUND(X,G2,V,B1,V)
2570 CALL SOUND(Y,E3,V,C1,V)
2580 CALL SOUND(X,D3,V,E1,V)
2590 CALL SOUND(X,C3,V,E1,V)
2600 CALL SOUND(X,B3,V,G1,V)
2610 CALL SOUND(X,A3,V,G1,V)
2620 CALL SOUND(Y,D3,V,B1,V)
2630 CALL SOUND(X,C3,V,D1,V)
2640 CALL SOUND(X,B3,V,D1,V)
2650 CALL SOUND(X,A3,V,G1,V)
2660 CALL SOUND(X,G2,V,G1,V)
2670 CALL SOUND(X,A3,V,C1,V)
2680 CALL SOUND(X,B3,V,C1,V)
2690 CALL SOUND(Y,D2,V,C1,V)
2700 CALL SOUND(Y,FS2,V,D1,V)
2710 CALL SOUND(Y,G2,V,G1,V)
2720 CALL SOUND(Y,G2,V,D1,V)
2730 CALL SOUND(Y,G2,V,G1,V)
2740 PRINT "ENCORE UNE FOIS!"
2750 NEXT A
2760 END
```

Programme de "Frère Jacques"

```
1000 DATA 262,294,330,262
1010 DATA 262,294,330,262
1020 DATA 330,349,392,392
1030 DATA 330,349,392,392
1040 DATA 392,440,392,349,330
1050 DATA 330,262,262
1060 DATA 392,440,392,349,330
1070 DATA 330,262,262
1080 DATA 262,262,196,196,262,262,262
1090 DATA 262,262,196,196,262,262,262
1100 FOR I=1 TO 45
1110 READ A
1120 CALL SOUND(500,A,0)
1130 NEXT I
```

Le dernier robot

Edourd Jouan

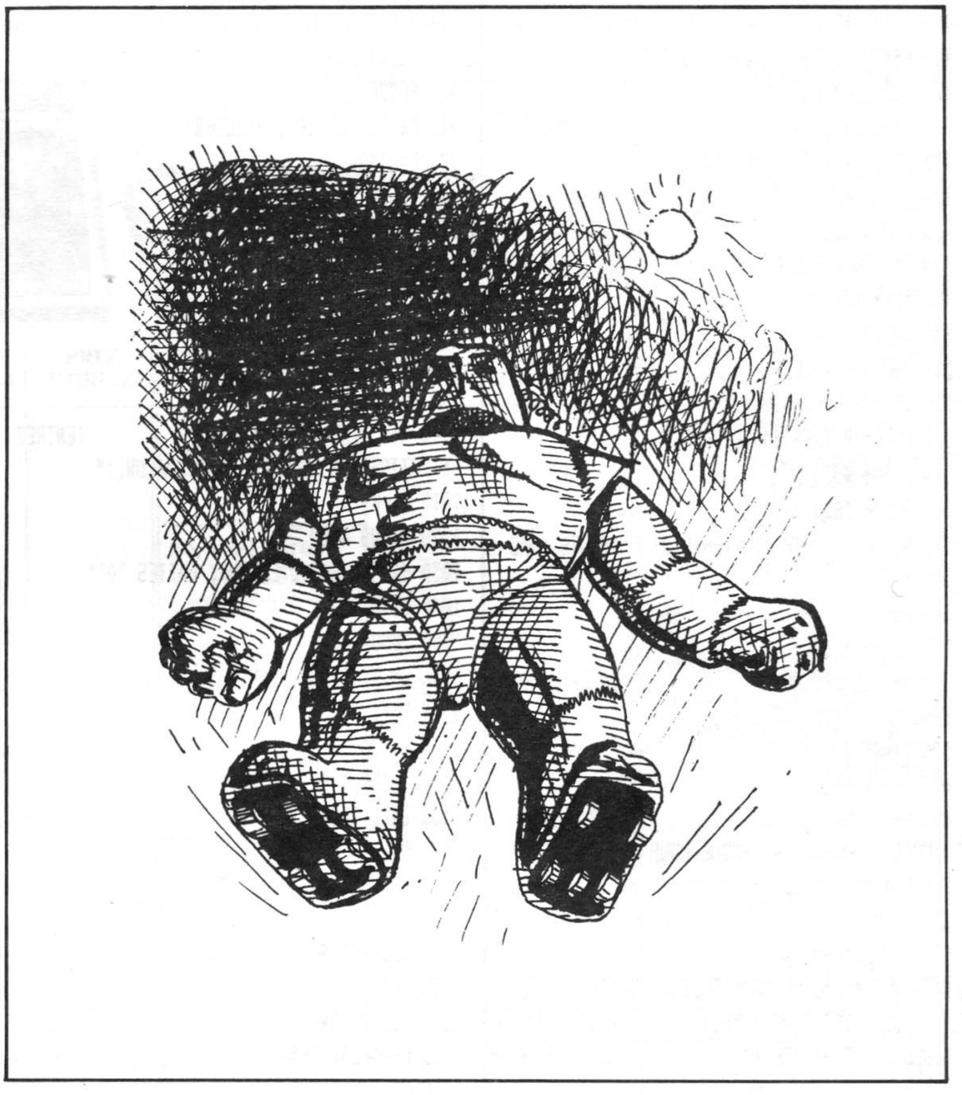
Le dernier robot est un programme de jeu qui nous a été gracieusement donné par le club TISOFT-HOME de Bruxelles. Il est beaucoup plus spectaculaire par ses qualités graphiques que par ses possibilités stratégiques. La règle est inspirée de celle du célèbre jeu de Marienbad : on dispose plusieurs lignes contenant chacune un nombre donné d'allumettes ; chacun des deux joueurs enlève dans une ligne autant d'objets qu'il le veut. Le perdant est celui qui enlève la dernière allumette.

Dans ce programme de jeu, il est possible de jouer à deux, ou seul contre l'ordinateur. Mais ce qui est le plus intéressant, c'est la représentation du jeu qui est faite par l'ordinateur. Les allumettes y sont remplacées par des robots admirablement dessinés, en plusieurs couleurs. En outre, la règle traditionnelle est modifiée dans ce jeu : il y a une limite (de 3 unités) au nombre de robots que l'on peut enlever en un coup. Ils sont cinq par ligne, plus un sur le côté gauche qui exécute les cinq autres.

Les choses se déroulent de la façon suivante : le joueur indique le numéro de la ligne dans laquelle il veut abattre des robots. Le robot se trouvant à la gauche de cette ligne tend alors son bras droit, armé d'un pistolet-laser. Le joueur indique le nombre de robots qu'il souhaite abattre. L'exécuteur oscille alors la tête : de haut en bas pour marquer son accord, ou de gauche à droite s'il n'est pas d'accord (vous avez indiqué plus de trois robots ou un nombre supérieur à celui des robots encore "vivants" dans la ligne).

Lorsque l'accord de l'exécuteur est acquis, ce dernier tire avec son pistolet-laser et élimine le nombre de robots désirés dans la ligne indiquée.

Nous avons aussi beaucoup aimé l'exposé qui est fait de la règle du jeu : il s'agit d'un texte montré du doigt par un robot qui fait du bruit pour attirer notre attention chaque fois qu'un nouveau paragraphe apparaît à l'écran. En outre, à la fin du jeu, un autre robot (ou le même, qui sait ?) traite de "sot" le perdant, que ce soit le joueur, désigné par son nom, ou le TI-99.



En conclusion, nous dirons que c'est là un programme excellent pour les jeunes enfants : la représentation graphique est parfaite et la stratégie de l'ordinateur assez pauvre pour donner des chances à tous. Quelqu'un veut-il modifier le programme pour rendre sa stratégie optimale ?

```
50 REM =====================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET EDOUARD JOUAN
80 REM =====================
90 REM
100 CALL CLEAR
110 M$="UN PETIT MOMENT"
120 R1=9
130 C1=5
140 GOSUB 4180
150 M$="POUR FORMER LA TROUPE..."
160 C1=C1-3
170 GOSUB 4180
180 R=9
190 C=2
200 FOR Z=1 TO 42
210 READ A,B$
220 CALL CHAR(A,B$)
230 NEXT Z
240 GOSUB 3170
250 GOSUB 3780
260 M$="LE DERNIER ROBOT"
270 L=-4000
280 GOSUB 4160
290 C1=C1+2
300 GOSUB 4180
```

```
310 M$="ECRIT PAR JIM MULLER"
320 R1=R1+8
330 GOSUB 4180
340 M$="RICHARDSON, TEXAS"
350 R1=R1+10
360 L=-2250
370 GOSUB 4160
380 GOSUB 4180
390 GOSUB 3850
400 GOSUB 4310
410 M$="VOULEZ-VOUS LES REGLES?"
420 GOSUB 3780
430 L=-2250
440 GOSUB 4160
450 GOSUB 4180
460 GOSUB 3850
470 M$="FAITES O OU N"
480 R1=R1+8
490 GOSUB 4180
500 CALL KEY(0,K,S)
510 IF S=0 THEN 500
520 IF K=79 THEN 530 ELSE 900
530 GOSUB 4310
540 GOSUB 3850
550 M$="IL Y A TROIS LIGNES DE 5 ROBOTS CHAC
UNE."
560 R1=R1-6
570 GOSUB 4180
580 M$="LE JEU CONSISTE A      LAISSER L'ADV
ERSAIRE   TIRER LE DERNIER."
590 R1=R1-3
600 GOSUB 4180
610 M$="CHAQUE JOUEUR PEUT     TIRER JUSQU'A
 TROIS    ROBOTS A CHAQUE COUP   DAN
S UNE MEME LIGNE."
620 R1=R1+1
630 GOSUB 4180
640 M$="UN JOUEUR SEUL JOUE    CONTRE LE TI-
99. (IL   EST TOUT A FAIT        POS
SIBLE DE LE BATTRE!)"
650 R1=R1+5
660 GOSUB 4180
670 X=600
680 GOSUB 4280
690 GOSUB 4310
700 M$="POUR JOUER"
710 R1=R1-6
720 GOSUB 4180
730 CALL HCHAR(R1-5,10,122,LEN(M$))
740 X=50
750 GOSUB 4280
760 M$="INDIQUEZ LA LIGNE, PUISLE NOMBRE DE
ROBOTS. LEROBOT DE GAUCHE SE     CHA
RGE DES HAUTES      OEUVRES..."
770 R1=R1-4
780 GOSUB 4180
790 M$="1,2,OU 3 INDIQUENT LE  NOMBRE DE ROB
OTS"
800 R1=R1+1
810 GOSUB 4180
820 M$="IL VALIDE VOTRE CHOIX, PUIS IL L'EXE
CUTE."
830 R1=R1+4
840 GOSUB 4180
850 M$="LA LIGNE DU BAS DE     L'ECRAN VOUS
INDIQUE   LA MARCHE DU JEU."
860 R1=R1+8
870 GOSUB 4180
880 X=500
890 GOSUB 4280
900 GOSUB 4310
910 M$="1, 2, OR 3 JOUEURS?"
920 L=-2000
930 GOSUB 3780
940 GOSUB 4160
950 GOSUB 4180
960 GOSUB 3850
970 GOSUB 3280
980 GOSUB 4480
990 GOSUB 4310
1000 M$="NOMS DES JOUEURS :           (ENTREZ
 6 LETTRES OU ESPACES POUR CHACUN.)"
1010 R1=R1-6
1020 GOSUB 4180
1030 M$="EN CAS D'ERREUR, FAITES '0'"
1040 R1=R1+12
1050 GOSUB 4180
1060 ON N1 GOSUB 4560,4630,4690
1070 P=N1
1080 X=120
1090 GOSUB 4280
1100 CALL CLEAR
1110 R=1
1120 GOSUB 3040
1130 R=9
1140 GOSUB 3040
1150 R=17
1160 GOSUB 3040
1170 M$="PLAYER:"
1180 R1=24
1190 C1=2
1200 GOSUB 4180
1210 A=5
1220 B=5
1230 CC=5
1240 T=15
1250 ON P GOSUB 1280,1280,1400
1260 GOSUB 1310
1270 GOTO 1260
1280 RANDOMIZE
1290 G2=INT(2*RND)+1
1300 ON G2 GOTO 1310,1350
1310 P$=P1$
1320 IF T=1 THEN 2780
1330 GOSUB 5070
1340 GOSUB 1510
1350 P$=P2$
1360 IF T=1 THEN 2780
1370 GOSUB 5070
1380 GOSUB 1510
1390 RETURN
1400 RANDOMIZE
1410 G3=INT(3*RND)+1
1420 ON G3 GOSUB 1310,1350,1450
1430 GOSUB 1450
1440 GOTO 1430
1450 P$=P3$
1460 IF T=1 THEN 2780
1470 GOSUB 5070
1480 GOSUB 1510
1490 GOSUB 1310
1500 RETURN
1510 IF P$="TI99/4" THEN 1520 ELSE 1560
1520 N=N+1
1530 IF N>3 THEN 1540 ELSE 1570
1540 N=1
1550 GOTO 1570
1560 GOSUB 4370
1570 C=1
1580 ON N GOTO 1590,1650,1710
1590 R=1
1600 D=A
1610 GOSUB 1770
1620 IF ZZ=0 THEN 1510
1630 A=D
1640 RETURN
1650 R=9
1660 D=B
1670 GOSUB 1770
1680 IF ZZ=0 THEN 1510
1690 B=D
1700 RETURN
1710 R=17
1720 D=CC
1730 GOSUB 1770
1740 IF ZZ=0 THEN 1510
1750 CC=D
1760 RETURN
1770 GOSUB 3780
1780 IF P$="TI99/4" THEN 1790 ELSE 1870
1790 IF T>12 THEN 1800 ELSE 1820
1800 N1=1
1810 GOTO 1880
1820 IF T-9>0 THEN 1830 ELSE 1850
1830 N1=T-9
1840 GOTO 1880
1850 N1=4-N1
1860 GOTO 1880
1870 GOSUB 4440
1880 R1=R
1890 GOSUB 3510
1900 IF T-N1<=0 THEN 1920
1910 IF D<N1 THEN 1920 ELSE 2040
1920 GOSUB 3560
1930 GOSUB 3850
```

Suite page 48

Suite de la page 46

```
1940 IF P$="TI99/4" THEN 1950 ELSE 2020
1950 IF A)=N1 THEN 1980
1960 IF B)=N1 THEN 1980
1970 IF CC)=N1 THEN 1980 ELSE 2010
1980 N1=4-N1
1990 ZZ=0
2000 RETURN
2010 N1=3
2020 ZZ=0
2030 RETURN
2040 GOSUB 3670
2050 GOSUB 3510
2060 R1=R
2070 ON N1 GOTO 2080,2100,2120
2080 ON D GOSUB 2320,2290,2260,2230,2200
2090 GOTO 2130
2100 ON D GOSUB 2760,2500,2450,2400,2350
2110 GOTO 2130
2120 ON D GOSUB 2760,2760,2690,2620,2550
2130 GOSUB 3280
2140 GOSUB 3850
2150 D=D-N1
2160 T=T-N1
2170 ZZ=1
2180 ZZ=1
2190 RETURN
2200 C1=8
2210 GOSUB 4020
2220 RETURN
2230 C1=13
2240 GOSUB 4020
2250 RETURN
2260 C1=18
2270 GOSUB 4020
2280 RETURN
2290 C1=23
2300 GOSUB 4020
2310 RETURN
2320 C1=28
2330 GOSUB 4020
2340 RETURN
2350 C1=8
2360 GOSUB 4020
2370 C1=13
2380 GOSUB 4020
2390 RETURN
2400 C1=13
2410 GOSUB 4020
2420 C1=18
2430 GOSUB 4020
2440 RETURN
2450 C1=18
2460 GOSUB 4020
2470 C1=23
2480 GOSUB 4020
2490 RETURN
2500 C1=23
2510 GOSUB 4020
2520 C1=28
2530 GOSUB 4020
2540 RETURN
2550 C1=8
2560 GOSUB 4020
2570 C1=13
2580 GOSUB 4020
2590 C1=18
2600 GOSUB 4020
2610 RETURN
2620 C1=13
2630 GOSUB 4020
2640 C1=18
2650 GOSUB 4020
2660 C1=23
2670 GOSUB 4020
2680 RETURN
2690 C1=18
2700 GOSUB 4020
2710 C1=23
2720 GOSUB 4020
2730 C1=28
2740 GOSUB 4020
2750 RETURN
2760 REM BLANK
2770 RETURN
2780 CALL CLEAR
2790 M$=P$
2800 R1=4
2810 C1=8
2820 GOSUB 4180
2830 R=8
2840 C=14
2850 GOSUB 3260
2860 CALL VCHAR(R+4,C,32,3)
2870 MSG$="TU AS PERDU, SOT!"
2880 R1=18
2890 C1=8
2900 FOR W=1 TO LEN(MSG$)
2910 CALL HCHAR(R+4,C,139)
2920 CALL SOUND(-550,110,3,-7,0)
2930 CALL HCHAR(R1,C1,ASC(SEG$(MSG$,W,1)))
2940 C1=C1+1
2950 NEXT W
2960 GOSUB 3260
2970 M$="UNE TOUCHE POUR REJOUER."
2980 R1=24
2990 C1=3
3000 GOSUB 4180
3010 CALL KEY(0,K,S)
3020 IF S=0 THEN 3010
3030 IF K()0 THEN 1100
3040 C=1
3050 GOSUB 3260
3060 C=9
3070 GOSUB 3260
3080 C=14
3090 GOSUB 3260
3100 C=19
3110 GOSUB 3260
3120 C=24
3130 GOSUB 3260
3140 C=29
3150 GOSUB 3260
3160 RETURN
3170 CALL COLOR(9,16,1)
3180 CALL COLOR(10,6,1)
3190 CALL COLOR(11,7,1)
3200 CALL COLOR(12,2,1)
3210 CALL COLOR(13,16,1)
3220 CALL COLOR(14,7,1)
3230 CALL COLOR(15,6,1)
3240 CALL COLOR(16,16,6)
3250 CALL CLEAR
3260 GOSUB 3280
3270 GOTO 3330
3280 CALL HCHAR(R,C+1,96)
3290 CALL HCHAR(R,C+2,97)
3300 CALL HCHAR(R+1,C+1,98)
3310 CALL HCHAR(R+1,C+2,99)
3320 RETURN
3330 CALL HCHAR(R+2,C,104)
3340 CALL HCHAR(R+2,C+1,105,2)
3350 CALL HCHAR(R+2,C+3,106)
3360 CALL HCHAR(R+3,C,107)
3370 CALL HCHAR(R+3,C+1,108)
3380 CALL HCHAR(R+3,C+2,109)
3390 CALL HCHAR(R+3,C+3,110)
3400 CALL HCHAR(R+4,C,112)
3410 CALL HCHAR(R+4,C+1,113)
3420 CALL HCHAR(R+4,C+2,114)
3430 CALL HCHAR(R+4,C+3,115)
3440 CALL HCHAR(R+5,C,116)
3450 CALL HCHAR(R+5,C+1,117)
3460 CALL HCHAR(R+5,C+2,118)
3470 CALL HCHAR(R+5,C+3,119)
3480 CALL HCHAR(R+6,C+1,120)
3490 CALL HCHAR(R+6,C+2,121)
3500 RETURN
3510 CALL HCHAR(R,C+1,100)
3520 CALL HCHAR(R,C+2,101)
3530 CALL HCHAR(R+1,C+1,102)
3540 CALL HCHAR(R+1,C+2,103)
3550 RETURN
3560 GOSUB 3280
3570 FOR Y=1 TO 3
3580 GOSUB 3510
3590 GOSUB 3620
3600 NEXT Y
3610 RETURN
3620 CALL HCHAR(R,C+1,132)
3630 CALL HCHAR(R,C+2,133)
3640 CALL HCHAR(R+1,C+1,134)
3650 CALL HCHAR(R+1,C+2,135)
3660 RETURN
3670 FOR Y=1 TO 3
3680 GOSUB 3280
```

```
3690 CALL HCHAR(R,C+1,32,2)
3700 CALL HCHAR(R+1,C+1,130)
3710 CALL HCHAR(R+1,C+2,131)
3720 CALL HCHAR(R+2,C+1,152)
3730 CALL HCHAR(R+2,C+2,153)
3740 CALL HCHAR(R+2,C+1,105,2)
3750 NEXT Y
3760 GOSUB 3280
3770 RETURN
3780 CALL HCHAR(R+2,C+3,144)
3790 CALL HCHAR(R+2,C+4,145)
3800 CALL VCHAR(R+3,C+3,32,3)
3810 CALL HCHAR(R+2,C+5,136)
3820 CALL HCHAR(R+2,C+6,137)
3830 GOSUB 3510
3840 RETURN
3850 CALL HCHAR(R+2,C+4,32,3)
3860 CALL HCHAR(R+2,C+3,106)
3870 CALL HCHAR(R+3,C+3,110)
3880 CALL HCHAR(R+4,C+3,115)
3890 CALL HCHAR(R+5,C+3,119)
3900 GOSUB 3280
3910 RETURN
3920 FOR I=C+7 TO C1
3930 CALL HCHAR(R+2,I,138)
3940 L=-500
3950 GOSUB 4170
3960 NEXT I
3970 CALL SOUND(1000,-7,0)
3980 FOR I=C+7 TO C1
3990 CALL HCHAR(R+2,I,32)
4000 NEXT I
4010 RETURN
4020 GOSUB 3920
4030 CALL SOUND(-200,110,1,-7,0)
4040 CALL VCHAR(R1+2,C1+1,139,4)
4050 CALL VCHAR(R1+2,C1+1,32,4)
4060 CALL SOUND(-200,110,1,-7,0)
4070 CALL VCHAR(R1,C1+2,139,7)
4080 CALL VCHAR(R1,C1+2,32,7)
4090 CALL SOUND(-200,110,1,-7,0)
4100 CALL VCHAR(R1,C1+3,139,7)
4110 CALL VCHAR(R1,C1+3,32,7)
4120 CALL SOUND(-200,110,1,-7,0)
4130 CALL VCHAR(R1+2,C1+4,139,4)
4140 CALL VCHAR(R1+2,C1+4,32,4)
4150 RETURN
4160 CALL SOUND(L,1760,1,-7,0)
4170 RETURN
4180 FOR A1=1 TO LEN(M$)
4190 CALL HCHAR(R1,C1,ASC(SEG$(M$,A1,1)))
4200 C1=C1+1
4210 IF C1>32 THEN 4220 ELSE 4240
4220 R1=R1+1
4230 C1=10
4240 NEXT A1
4250 R1=11
4260 C1=10
4270 RETURN
4280 FOR DE=1 TO X
4290 NEXT DE
4300 RETURN
4310 REM CLEAR SCREEN
4320 FOR R2=5 TO 23
4330 C1=10
4340 CALL HCHAR(R2,C1,32,23)
4350 NEXT R2
4360 RETURN
4370 M$="LIGNE ?"
4380 R1=24
4390 C1=17
4400 GOSUB 4180
4410 GOSUB 4510
4420 N=K-48
4430 RETURN
4440 M$="NOMBRE?"
4450 R1=24
4460 C1=17
4470 GOSUB 4180
4480 GOSUB 4510
4490 N1=K-48
4500 RETURN
4510 CALL KEY(0,K,S)
4520 IF S=0 THEN 4510
4530 IF K<49 THEN 4510
4540 IF K>51 THEN 4510
4550 RETURN
4560 GOSUB 4780
4570 P2$="TI99/4"
4580 M$=P2$
4590 R1=R1+4
4600 C1=C1+11
4610 GOSUB 4180
4620 RETURN
4630 GOSUB 4780
4640 R1=R1+4
4650 GOSUB 4870
4660 P2$=P$
4670 RETURN
4680 IF N1=1 THEN 4690 ELSE 4760
4690 GOSUB 4630
4700 M$="JOUEUR 3:"
4710 R1=R1+4
4720 C1=10
4730 GOSUB 4180
4740 R1=R1+8
4750 GOSUB 4870
4760 P3$=P$
4770 RETURN
4780 M$="JOUEUR 1:"
4790 GOSUB 4180
4800 GOSUB 4870
4810 P1$=P$
4820 M$="JOUEUR 2:"
4830 R1=R1+4
4840 C1=10
4850 GOSUB 4180
4860 RETURN
4870 FOR V=1 TO 6
4880 CALL KEY(0,K,S)
4890 IF S=0 THEN 4880
4900 C1=20+V
4910 CALL HCHAR(R1,C1,K)
4920 IF K=48 THEN 4950 ELSE 4970
4930 X=15
4940 GOSUB 4280
4950 CALL HCHAR(R1,C1-V,32,7)
4960 GOTO 4870
4970 NEXT V
4980 C1=20
4990 CALL GCHAR(R1,C1+1,K1)
5000 CALL GCHAR(R1,C1+2,K2)
5010 CALL GCHAR(R1,C1+3,K3)
5020 CALL GCHAR(R1,C1+4,K4)
5030 CALL GCHAR(R1,C1+5,K5)
5040 CALL GCHAR(R1,C1+6,K6)
5050 P$=CHR$(K1)&CHR$(K2)&CHR$(K3)&CHR$(K4)&
CHR$(K5)&CHR$(K6)
5060 RETURN
5070 FOR X=1 TO 6
5080 R1=24
5090 C1=8+X
5100 CALL HCHAR(R1,C1,ASC(SEG$(P$,X,1)))
5110 NEXT X
5120 RETURN
5130 DATA 96,0F1F3F7F7F7F6060,97,F0F8FCFEFEF
E0606,98,61637F7F3F381F0F,99,86C6FEF
EFC1CF8F0
5140 DATA 100,0F1F3F7F7F7F7F7F,101,F8FCFCFCF
CF2F2F2,102,7F7F7F7F3F3F1F0F,103,F3F
7FFFFFFC0F8F0
5150 DATA 104,3F7FFFFFFFFFFFFB,105,FFFFFFFF
FFFFFFF,106,FCFEFFFFFFFFFFDF,107,F3F
1F1F0F0F0F000
5160 DATA 108,FFFFFFFFFF7F7F00,109,FFFFFFFF
FFEFE00,110,CF8F8F0F0F0F0F00,111,7F7
F7F7F1F0F0703
5170 DATA 112,F0F0F0F0F0F1F1F3,113,3F7F7F7FF
FFFFFFF,114,FCFEFEFFFFFFFFFF,115,0F0
F0F0F0F8F8FC0
5180 DATA 116,60E0C07000000000,117,7F7F3E3E3
E1C1C00,118,FEFE7C7C7C383800,119,060
6030E00000000
5190 DATA 120,1C1C1C1C1C1C3E3E,121,383838383
8387C7C,122,FF00000000000000
5200 DATA 130,0F1F3F7F7F7F7F7F,131,F0F8FCFEF
EFEFEFE,132,1F3F3F3F3F4F4F4F,133,F0F
8FCFEFEFEFEFE
5210 DATA 134,CFEFFFFFFF031F0F,135,FEFEFEFEF
CF8F0E0
5220 DATA 136,7F7F7F7F00000000,137,7F7060400
0000000,138,FF00000000000000,139,249
255BE7CB7245B
5230 DATA 140,FF07030100000000
5240 DATA 144,FFFFFFFF00000000,145,FFFFFFFF0
0000000
5250 DATA 152,60617F3F1F000000,153,0686FEFCF
8000000
```

Cherchez l'ancêtre

Robert Peltier

Voulez-vous rencontrer Louis XVI, Clémenceau, Voltaire ?... Désirez-vous remonter le temps jusqu'à la prise de la Bastille ou préférez-vous assister à la première représentation d'un film parlant ? Si c'est le cas, laissez-vous aller au fond de votre fauteuil et lancez le programme !

Non, restons sérieux. Je ne vous propose pas de transformer votre ordinateur en machine à explorer le temps. Toutefois, ce petit programme très simple, écrit en TI Basic, vous permettra de réviser vos connaissances historiques tout en vous amusant. Après l'affichage du titre, l'ordinateur vous annonce à quel évènement historique vous pourrez "assister"... à condition d'en retrouver la date. Rassurez-vous, l'ordinateur vous guidera dans la recherche de cette date; de plus, il se montre bon prince et tolère quelques années d'écart.

Mais le jeu ne s'arrête pas là : une fois la date exacte de l'évènement retrouvée, il s'agit pour vous de rencontrer l'un des glorieux ancêtres dont l'ordinateur vous affiche aimablement la liste, ce qui suppose bien entendu que cet ancêtre soit contemporain à l'évènement.

Facile, dites-vous ? Sans doute, si vous venez d'assister à la prise de la Bastille, pouvez-vous sans trop de risque vous proposer de rencontrer Louis XVI... Mais tous les cas de figure ne sont pas aussi simples. De plus, la version du programme qui vous est proposée ici n'est que l'une des versions possibles du jeu.

Ainsi, vous pouvez sans difficulté apporter quelques modifications intéressantes. Pour commencer, essayez par exemple d'agir sur la difficulté du programme en diminuant la marge de tolérance aux erreurs de date. Cette tolérance est de 5 ans dans la version que nous vous proposons ici, comme vous le constaterez dans les lignes 770 et 780 ; vous pouvez par exemple écrire :

780 IF DOF>2 THEN 790 ELSE 820

ce qui réduit la tolérance à deux années au lieu de 5.

En outre, vous pourrez modifier la liste des évènements historiques en remplaçant les lignes de DATA qui figurent au début du programme, de 130 à 200.

Enfin, rien ne vous interdit d'allonger cette liste (la version actuelle n'occupe que 5K octets de RAM). L'opération est simple : il suffit d'intercaler de nouvelles lignes de DATA entre les lignes existantes. Mais prenez toutefois la précaution d'ajouter pour chaque nouvel évènement une unité supplémentaire à la valeur indiquée dans la ligne 410 :

410 GG= INT(19*RND)+1

devient ainsi :

410 GG= INT(20*RND)+1

après la "création" d'un évènement supplémentaire.

Un autre moyen de rendre ce jeu plus attrayant consiste à jouer à plusieurs. Dans ce cas, je vous propose les règles suivantes :

– Tout d'abord effacez du programme la liste des personnages historiques et confectionnez à l'aide de petits rectangles de bristol des cartes au format cartes à jouer. Sur ces cartes est reporté le nom d'un personnage du jeu.

– On tire au sort le joueur qui commence et on répartit les cartes entre tous les joueurs. Pour le premier tour, aucun joueur n'a le droit de regarder les cartes qu'il a tirées.

– Le joueur qui a la main annonce son contrat (nombre de points qu'il désire marquer). Les autres joueurs peuvent lui reprendre la main en proposant un contrat plus élevé... à leurs risques et périls. Celui des joueurs qui a la main joue alors seul.

– les points de chaque joueur sont cumulés et la partie s'achève soit lorsqu'un joueur a atteint un nombre de points déterminé à l'avance, soit lorsqu'un joueur n'a plus de carte.

A l'intention des débutants (je veux dire des grands débutants... parce que je ne suis moi-même qu'un débutant), je préciserai un peu la recette de ce programme qui est en gros celle d'un programme d'enseignement assisté par ordinateur. Son intérêt est de pouvoir être adapté à l'écriture de nombreux jeux très simples :

1 - Dans un certain nombre de lignes DATA, les questions et les réponses correspondantes sont regroupées à la suite les unes des autres. Par exemple :

10 DATA "Quelle est la capitale de l'Italie ?","ROME","Combien font 2+2","4", etc...

2 - La lecture des questions peut se faire en suivant l'ordre d'écriture des lignes de DATA; elle peut se faire également de façon aléatoire. Dans ce dernier cas, reportez-vous à la solution adoptée dans le programme "Retrouvez l'ancêtre" et à votre manuel de programmation pour l'utilisation de RESTORE.

Dans le cas de la lecture dans l'ordre des lignes de DATA, l'instruction 100 READ A$,B$ permet à l'ordinateur de lire deux variables :

A$ est la chaîne de caractères représentant la question ;

B$ est la réponse correspondante.

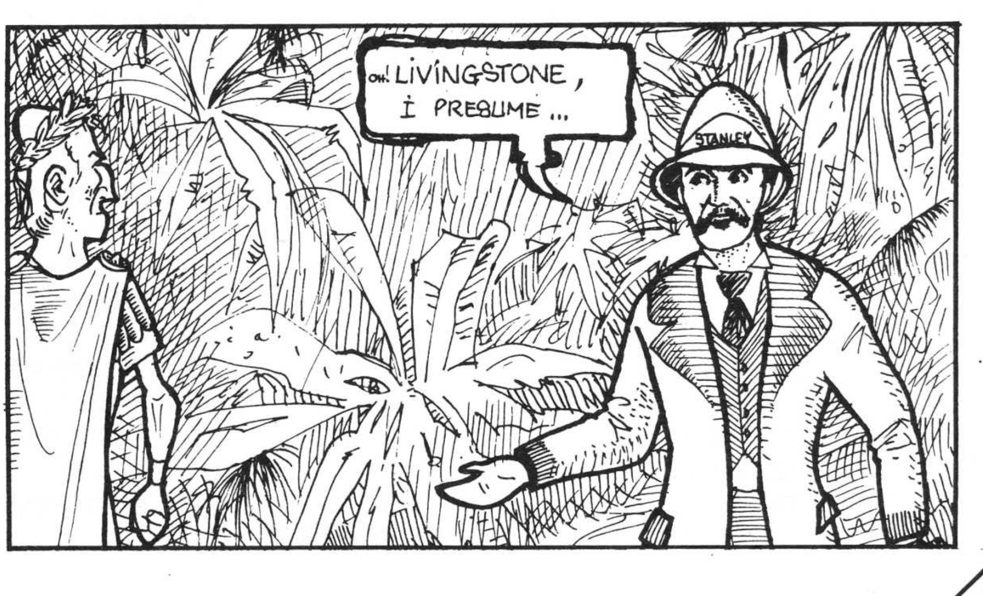
La structure générale de ce programme est reprise dans les quelques lignes ci-dessous qui ne constituent pas elles-mêmes un programme complet mais peuvent vous servir de point de départ pour l'élaboration de jeux basés sur les mêmes principes :

110 PRINT A$ permet l'affichage de la question (n'ajoutez pas ",B$", ce serait tricher !)

120 INPUT C$ vous invite à introduire votre réponse que le programme compare à la bonne réponse (B$).

```
200 IF B$=C$ THEN 210 ELSE
    230
210 PRINT "BRAVO"
220 GOTO 100
230 PRINT "C'EST FAUX !"
240 GOTO 120
```

Elémentaire, mon cher lecteur !



Programme "cherchez l'ancêtre"

```
50 REM =====================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET ROBERT PELTIER
80 REM =====================
90 REM
100 REM RETROUVEZ L ANCETRE  OCCUPE 4,6
Koctets DANS CETTE VERSION .IL EST DONC
PARFAITEMENT POSSIBLE D'ALLONGER DE FACO
N SUBSTANTIELLE
110 REM LA LISTE DES EVENEMENTS ET LA LI
STE DES PERSONNAGES
120 REM 27/2/83 R.PELTIER
130 DATA "LA PRISE DE LA BASTILLE",1789,
"L'INVENTION DE LA PHOTO SUR   VERRE PAR
 N.NIEPCE",1822,"LA CONSTITUTION DES ETA
TS UNIS DU BRESIL",1889
140 DATA "LA PROJECTION DU PREMIER FILM
 PARLANT",1927,"LA PREMIERE REPRESENTATI
ON   PUBLIQUE DU CINEMATOGRAPHE    LUMIE
RE",1895
150 DATA "LA PARUTION DU PREMIER TOME DU
 CAPITAL",1867,"LA PRISE DU POUVOIR PAR
BENITO MUSSOLINI",1933,"LA NAISSANCE DE
A.HITLER",1889
160 DATA "LA MORT DE LENINE",1924
170 DATA "LA PARUTION DU MANIFESTE DU PA
RTI COMMUNISTE",1848,"LA BATAILLE D'HERN
ANI",1830,"LA PUBLICATION DU CONTRAT   S
OCIAL",1762
180 DATA "LA PROCLAMATION DE L INDEPEN-D
ANCE DES U.S.A.",1776,"LA COMMUNE DE PAR
IS",1871,"LA BATAILLE DE WATERLOO",1815
190 DATA "LA FONDATION DU JOURNAL L'  HU
MANITE",1904,"LA MORT DE MOLIERE",1673,"
LA MORT DE MAZARIN",1661
200 DATA "LA CESSION DE LA SAVOIE ET  DE
 NICE",1860,"LA DECOUVERTE DE LA PENICIL
LINE",1929
210 DATA " ",0
220 CALL CLEAR
230 CALL SCREEN(4)
240 PRINT "   RETROUVEZ L'ANCETRE"
250 PRINT "   -------------------        "
260 PRINT
270 PRINT
280 PRINT
290 PRINT
300 FOR DEL=1 TO 800
310 NEXT DEL
320 CALL CLEAR
330 CALL SCREEN(11)
340 F=48
350 DEF TOTPENA=F-48
360 CALL COLOR(4,16,2)
370 CALL COLOR(3,16,2)
380 PRINT "JE VOUS PROPOSE DE REMONTER L
E TEMPS JUSQU'A"
390 RANDOMIZE
400 RESTORE 130
410 GG=INT(19*RND)+1
420 FOR LEC=1 TO GG
430 READ EV$,D
440 NEXT LEC
450 RESTORE
460 PRINT EV$
470 PRINT
480 PRINT
490 PRINT
500 PRINT "VOUS DEVREZ"
510 PRINT "1o_TROUVER LA DATE EXACTE   D
E CET EVENEMENT"
520 PRINT "2o_RENCONTRER A CETTE DATE  A
U MOINS UN PERSONNAGE DONT VOICI LA LIST
E "
530 PRINT
540 FOR DEL=1 TO 450
550 NEXT DEL
560 GOSUB 1680
570 PRINT
580 PRINT "TENTEZ DE MARQUER LE MAXIMUM
DE POINTS ; MAIS ATTENTION : UN CONTRAT
ELEVE NE VOUS  AUTORISE QUE PEU D ERREUR
S "
590 PRINT "PAR EXEMPLE POUR UN CONTRAT D
E 10 POINTS VOUS N'AVEZ    DROIT QU'A 2
ERREURS"
600 INPUT "COMBIEN DE POINTS VOULEZ    V
OUS TENTER (DE 1A 10)":T
610 PRINT "O.K. VOUS AVEZ DROIT A";(12-T
);"ERREURS  "
620 FOR DEL=1 TO 450
630 NEXT DEL
640 PRINT "  QUELLE EST LA DATE DE ";EV$
650 INPUT W
660 PRINT
670 PRINT
680 FOR M=1 TO L
690 NEXT M
700 IF D<>W THEN 710 ELSE 870
710 PRINT "NON, CELA S EST PASSE PLUS"
720 IF D>W THEN 730 ELSE 760
730 PRINT "TARD"
740 LET DIF=D-W
750 IF DIF<5 THEN 790 ELSE 810
760 PRINT "TOT"
770 LET DOF=W-D
780 IF DOF<5 THEN 790 ELSE 810
790 PRINT "MAIS ADMETTONS...VOUS N EN  E
TES PAS LOIN.              LA DATE EXACT
E EST"
800 GOTO 880
810 IF T+TOTPENA>10 THEN 1520 ELSE 820
```

```
820 PRINT "ESSAYEZ ENCORE"
830 F=F+1
840 CALL HCHAR(3,30,F)
850 RESTORE
860 GOTO 640
870 PRINT "EXACT"
880 CALL HCHAR(3,30,F)
890 PRINT
900 PRINT
910 FOR DELAI=1 TO 1000
920 NEXT DELAI
930 DATE=D
940 CALL CLEAR
950 CALL SCREEN(6)
960 PRINT
970 PRINT TAB(10);DATE
980 PRINT
990 PRINT SEG$(EV$,3,40)
1000 PRINT
1010 PRINT
1020 FOR DELAI=1 TO 300
1030 NEXT DELAI
1040 PRINT
1050 PRINT
1060 CALL HCHAR(3,30,F)
1070 PRINT
1080 PRINT "JE VOUS RAPPELLE LA LISTE
DES PERSONNAGES"
1090 GOSUB 1680
1100 PRINT "QUEL PERSONNAGE VOULEZ-VOUS
RENCONTRER A CETTE DATE "
1110 INPUT A$
1120 RESTORE 1610
1130 READ PAR$,DEB,FIN
1140 IF PAR$="FIN" THEN 1150 ELSE 1180
1150 RESTORE 1610
1160 PRINT "JE NE CONNAIS PAS CE ";A$
1170 GOTO 1080
1180 IF A$=PAR$ THEN 1190 ELSE 1130
1190 N=DEB
1200 M=FIN
1210 IF DATE<N THEN 1220 ELSE 1280
1220 PRINT
1230 PRINT "DESOLE MAIS VOUS NE POUVEZ
PAS ENCORE RENCONTRER"
1240 PRINT A$
1250 PRINT "QUI NAITRA EN ";N
1260 F=F+1
1270 IF T+TOTPENA>10 THEN 1520 ELSE 1040
1280 IF DATE>M THEN 1290 ELSE 1350
1290 PRINT
1300 PRINT "DESOLE MAIS VOUS NE POUVEZ
PLUS RENCONTRER"
1310 PRINT A$
1320 PRINT "QUI EST MORT EN ";M
1330 F=F+1
1340 IF T+TOTPENA>10 THEN 1520 ELSE 1040
1350 X=DATE-N
1360 PRINT
1370 PRINT A$
1380 PRINT "A CETTE ANNEE"
1390 PRINT X
1400 PRINT "ANS"
1410 FOR DELAI=1 TO 500
1420 NEXT DELAI
1430 CALL HCHAR(3,30,F)
1440 FOR DELAI=1 TO 2000
1450 NEXT DELAI
1460 PRINT "PENALITE";TOTPENA
1470 IF T+TOTPENA<11 THEN 1480 ELSE 1520
1480 PRINT "VOUS AVEZ REMPLI VOTRE CONTR
AT. VOUS MARQUEZ "
1490 PRINT T
1500 PRINT "POINTS"
1510 GOTO 1530
1520 PRINT "VOUS N AVEZ PAS REMPLI VOTRE
 CONTRAT VOUS PERDEZ 10 POINTS"
1530 FOR DELAI=1 TO 800
1540 NEXT DELAI
1550 CALL CLEAR
1560 CALL SCREEN(16)
1570 INPUT "VOULEZ VOUS UNE AUTRE PARTIE
(O/N)":Q$
1580 IF Q$="O" THEN 320 ELSE 1590
1590 PRINT "O.K. SALUT"
1600 STOP
1610 DATA "VOLTAIRE",1694,1778,"CLEMENCE
AU",1841,1929,"LOUIS XVI",1754,1793
1620 DATA "PARMENTIER",1737,1813,"ZEPPEL
IN",1838,1917,"WASHINGTON",1732,1799,"WA
GNER",1813,1883
1630 DATA "BEETHOVEN",1770,1827,"LENINE"
,1870,1927,"LA FAYETTE",1757,1834,"MOZAR
T",1756,1791
1640 DATA "CASANOVA",1725,1798,"ROSA LUX
EMBOURG",1870,1919,"SARTRE",1905,1980,"E
INSTEIN",1879,1955,"VICTORIA Io",1818,19
01
1650 DATA "LOUVOIS",1641,1691,"CORNEILLE
",1606,1684,"LA FONTAINE",1621,1695,"HEM
INGWAY",1899,1961,"CATHERINE II",1729,17
96
1660 DATA "VAUBAN",1633,1707
1670 DATA "FIN",0,0
1680 PRINT "VOLTAIRE     CLEMENCEAU
LOUIS XVI    PARMENTIER       ZEPPELIN
WASHINGTON       "
1690 PRINT "WAGNER       BEETHOVEN
LENINE       LA FAYETTE       MOZART
CASANOVA         "
1700 PRINT "SARTRE       EINSTEIN
VICTORIA Io LOUVOIS          CORNEILLE
LA FONTAINE      "
1710 PRINT "HEMINGWAY    CATHERINE II
VAUBAN"
1720 FOR DEL=1 TO 800
1730 NEXT DEL
1740 RETURN
```

Gestion Personnelle de Fichiers

Didier Bicking

La Gestion Personnelle de Fichiers est un module commercialisé par Texas Instruments pour le TI-99. Comme son nom l'indique, son objectif est de vous permettre de constituer, analyser et conserver des fichiers d'une taille limitée, adaptée aux besoins domestiques. Je me suis procuré ce module et je l'ai testé pour vous.

Lorsque le système se met en route, la première chose qui est demandée est la date. Toutes les opérations sur les fichiers seront datées, du moins sur les listings que je pourrai en faire. Il ne sera par exemple pas possible, sans manipulation, de récupérer un fichier donné tel qu'il se présentait à une date déterminée sauf si on a pris le soin de le sauvegarder dans ce but à cette date.

Le programme demande ensuite si je dispose d'une imprimante. Je réponds positivement et indique le nom de mon imprimante : "PIO". Le programme prévoit, outre les impressions normales d'éléments du fichier, la recopie sur imprimante de certains écrans. Nous verrons cela plus loin, mais remarquons à l'avance cet important détail qui peut se révéler très utile.

Afin de faire un test le plus complet possible, j'ai constitué successivement deux fichiers différents.

Les joueurs d'un club de tennis

Le premier travail consiste bien entendu à définir la structure de mon fichier. Chaque "article" du fichier est une page contenant un certain nombre d'éléments ayant chacun leurs caractéristiques propres. Je me propose de constituer, pour mon club de tennis, un fichier reprenant les nom et prénom de chaque joueur, accompagnés de son année de naissance, son sexe, son classement, un indice permettant de savoir s'il est à jour de ses cotisations, puis deux indices permettant de savoir s'il est inscrit d'une part au tournoi interne du club, d'autre part au tournoi "open".

J'ai donc défini chacun des éléments ainsi que vous pouvez le voir dans le tableau : le nom comporte au maximum 15 caractères (taille maximale admise par le programme), le prénom est sur huit caractères, etc... (tableau 2). La colonne DEC n'est pas utilisée ici : ce fichier ne contiendra en effet pas de nombres avec des décimales.

Une fois définie et confirmée la structure du fichier (le programme en demande confirmation et autorise des rectifications avant l'enregistrement définitif), j'ai demandé à revoir cette structure. C'est alors que m'ont été présentés successivement deux écrans qui peuvent être recopiés sur l'imprimante (tableaux 1 et 2). Le premier d'entre eux comporte une information essentielle : je ne pourrai constituer que 267 pages. Ce nombre est déterminé en fonction de la structure choisie : si l'on constitue un fichier comportant moins d'éléments, on disposera d'un plus grand nombre de pages, et inversement.

Il faut noter que le module n'utilise pas l'extension mémoire de 32K dont dispose mon système. D'un côté cela est regrettable car le nombre de pages est finalement peu important (dans la réalité, mon club de tennis compte 1800 membres...). En revanche, cela rend le module de Gestion Personnelle de Fichiers accessible à ceux qui ne possèdent que la version de base du TI-99, c'est-à-dire la console et un cordon de branchement pour un magnétophone.

```
        FILE STRUCTURE
 -------------------------------
        NAME:   TENNIS
        DATE:   4/29/83
 ITEMS/PAGE:    8
 PAGES USED:    10
 PAGES LEFT:    267
```

Tableau 1

```
        FILE STRUCTURE
 -------------------------------
  ITEM         TYPE WIDTH DEC
 1 NOM         CHAR  15    0
 2 PRENOM      CHAR  8     0
 3 NAISSANCE   INT   2     0
 4 SEXE        CHAR  1     0
 5 CLASST      INT   2     0
 6 COTIS.      INT   2     0
 7 TOURNOI 1   INT   1     0
 8 TOURNOI 2   INT   1     0
```

Tableau 2

```
        DISPLAY PAGE
 -------------------------------
  PAGE #?        4
  1 NOM          CLARTE
  2 PRENOM       RENAUD
  3 NAISSANCE    55
  4 SEXE         M
  5 CLASST       15
  6 COTIS.       83
  7 TOURNOI 1    1
  8 TOURNOI 2    1
```

Tableau 3

La structure du fichier étant déterminée, il m'a fallu remplir une à une les pages qui le constituent. Pour me simplifier les choses, j'ai fabriqué un indice pour les classements : le classement le plus fort (-30) est le classement 0, un joueur classé -15 est noté 1, etc. jusqu'au pauvre non classé que je suis et qui est noté 15.

J'ai ainsi rempli une dizaine de pages que j'ai ensuite voulu faire imprimer. Un choix à trois possibilités est offert par le programme pour les impressions :

- impression de chaque page séparément
- impression de toutes les pages sous forme d'un tableau
- impression d'éléments choisis dans chaque page, encore sous forme d'un tableau.

A titre d'exemple, j'ai fait imprimer une page (tableau 3), puis le fichier complet sous forme de tableau (tableau 4).

Le programme permet d'effectuer des tris. J'ai demandé, dans un premier temps, un tri alphabétique. Chaque opération de ce genre est effectuée sur un critère et un seul. Or mon fichier comporte des personnes ayant le même nom. Dans ce cas, le programme, comme tous les programmes de tri informatique, classe les éléments identiques dans l'ordre où il les a rencontrés. Pour obtenir un classement à la fois sur le nom et le prénom, il m'a donc fallu commencer par faire un tri sur le prénom, puis, immédiatement après, le tri sur le nom. On retrouve bien ainsi dans le tableau final (tableau 5) Anne BLASTE devant Jacques BLASTE (les noms sont directement issus de mon imagination).

Il est aussi possible d'isoler certains articles du fichier en fonction de critères choisis. Il est par exemple aisé d'établir la liste de ceux des mem-

bres qui ne sont pas à jour de leurs cotisations pour l'année 1983 (tableau 6). On pourrait aussi faire une liste des hommes non classés, etc...

Dans une liste, on n'a pas toujours besoin de tous les éléments que comporte chaque page. Le programme permet donc de n'éditer que les éléments choisis comme par exemple le nom, le prénom et le classement (tableau 7).

Gestion d'un catalogue

A titre de second exemple, j'ai imaginé le cas d'un commerçant, important des Etats Unis du matériel électronique. Il paye ses importations en dollars. Sur ce prix, il applique son coefficient de marge (par exemple 1.45) pour déterminer un prix de vente de base, toujours en dollars. Il effectue ensuite la conversion du prix de base en francs français. Enfin, il établit pour chaque article quatre prix de vente finaux correspondant chacun à une catégorie de client.

La structure de fichier étant déterminée (tableau 8), il dispose de 173 pages. Dans un premier temps, il ne remplit que les trois premiers éléments de chaque page, laissant à l'ordinateur le soin de calculer les autres (tableau 9). Pour cela, il remplit d'abord toutes les pages de son fichier puis demande à effectuer des transformations mathématiques. 18 opérations différentes sont possibles : transfert de valeur d'un élément à un autre, les quatre opérations de base, quelques opérations trigonométriques, l'extraction des logarithmes naturels ou décimaux, et même RND (le nombre aléatoire) dont on imagine mal l'utilité dans le cadre d'une gestion de fichier...

L'opération de multiplication se présente sous la forme : A =B⋆C. Le manipulateur choisit cette opération et indique ensuite quel élément est A. Le programme demande alors si B est une constante. En cas de réponse négative il demande quel élément est B, sinon il demande la valeur de B. Il en est de même enfin pour C. Il est ainsi possible d'établir les relations mathématiques qui relient entre eux les éléments d'une même page du fichier. Lorsque cela est fait, on lance l'exécution de ces transformations, c'est-à-dire que l'ordinateur effectue tous les calculs pour chacune des pages du fichier et y note les résultats obtenus.

Le commerçant peut alors faire éditer son catalogue de prix (tableau 10). Par la suite, si par exemple le cours du dollar change, il n'aura plus qu'à modifier en conséquence la relation entre le prix de base libellé en dollars et celui libellé en francs. Il y a cependant là un écueil important du programme : on ne peut modifier isolément une relation mathématique. Il faut en effet les modifier toutes ou aucune. En réalité, le programme considère qu'il n'y a pas de modification, mais que chaque opération consiste à établir un nouveau tissu de relations et, en conséquence, il annule toutes les précédentes. J'ai en

FILE: TENNIS
DATE: 5/21/83
TITLE: TENNIS

0	1	2	3	4	5	6	7	8
1	ARNAULT	PIERRE	39		7	83	1	1
2	BICKING	DIDIER	45	M	15	83	1	1
3	BICKING	CAROLINE	74	F	15	83	0	0
4	BLASTE	ANNE	56	F	13	83	1	0
5	BLASTE	JACQUES	44	M	9	82	0	1
6	CLARTE	RENAUD	55	M	15	83	1	1
7	MEURISSE	J.MARC	51	M	15	82	0	0
8	MEURISSE	MONIQUE	53	F	14	83	0	1
9	MICHEL	BERNARD	52	M	6	82	1	1
10	NOUJA	ANNICK	44	F	11	83	1	1

Tableau 5

FILE: TENNIS
DATE: 5/21/83
TITLE: TENNIS

0	1	2	3	4	5	6	7	8
1	BLASTE	JACQUES	44	M	9	82	0	1
2	MEURISSE	J.MARC	51	M	15	82	0	0
3	MICHEL	BERNARD	52	M	6	82	1	1

Tableau 6

FILE: TENNIS
DATE: 5/21/83
TITLE: TENNIS

INDEX
0 = PAGE #
1 = NOM
2 = PRENOM
3 = NAISSANCE
4 = SEXE
5 = CLASST
6 = COTIS.
7 = TOURNOI 1
8 = TOURNOI 2

0	1	2	3	4	5	6	7	8
1	MEURISSE	J.MARC	51	M	15	82	0	0
2	MICHEL	BERNARD	52	M	6	82	1	1
3	BLASTE	JACQUES	44	M	9	82	0	1
4	CLARTE	RENAUD	55	M	15	83	1	1
5	MEURISSE	MONIQUE	53	F	14	83	0	1
6	ARNAULT	PIERRE	39		7	83	1	1
7	BICKING	DIDIER	45	M	15	83	1	1
8	NOUJA	ANNICK	44	F	11	83	1	1
9	BICKING	CAROLINE	74	F	15	83	0	0
10	BLASTE	ANNE	56	F	13	83	1	0

Tableau 4

FILE: TENNIS
DATE: 5/21/83
TITLE: TENNIS

0	1	2	5
1	BLASTE	JACQUES	9
2	MEURISSE	J.MARC	15
3	MICHEL	BERNARD	6
4	BLASTE	ANNE	13
5	ARNAULT	PIERRE	7
6	CLARTE	RENAUD	15
7	BICKING	DIDIER	15
8	MEURISSE	MONIQUE	14
9	BICKING	CAROLINE	15
10	NOUJA	ANNICK	11

Tableau 7

conséquence choisi de noter dans chaque page un élément supplémentaire : le cours du dollar.

Les représentants n'auront certes pas besoin de connaître la composition complète du prix. Le commerçant peut alors éditer un catalogue de vente (tableau 11) ne comportant que les éléments indispensables à ceux à qui il est destiné.

Enfin, il est possible d'établir des statistiques sur chacun des éléments numériques composant le fichier. Le programme lit alors le même élément dans toutes les pages. Il en calcule la moyenne, l'écart-type, indique quels sont les montants maximum et minimum, la somme, le nombre de pages concernées et le nombre de données vides rencontrées. Une donnée vide est un élément non rempli dans une page (tableau 12).

Conclusion

La taille des fichiers qu'il est possible de constituer et utiliser avec ce programme le destinent effectivement à une gestion personnelle, bien que les exemples donnés soient orientés vers un certain professionalisme. Il n'en reste pas moins que ce programme est relativement puissant. J'ai notamment beaucoup apprécié la fonction de tri et la lecture statistique du fichier.

Il faut noter cependant une certaine lenteur, notamment lorsqu'il s'agit d'éditer un fichier sous forme de tableau. Le tableau 10, par exemple, a pris plus de deux minutes (pour dix lignes !). Cela provient de ce que le programme, à l'origine, a été écrit en Basic avant d'être intégré dans un module. Il prend le côté pratique du module (facilité de chargement), tout en gardant la lenteur d'exécution du langage Basic.

L'utilité d'un tel programme est incontestable et on ne peut faire une liste exhaustive des applications possibles. Citons à titre d'exemple :

- gestion d'une bibliothèque personnelle et des prêts de livres
- gestion d'une cave à vin
- annuaire personnel
- fichier de notes et moyennes pour un enseignant
- toutes sortes de fichiers scientifiques et statistiques
- inventaire domestique, avec actualisation de la valeur des objets
- gestion et évaluation de stocks
- établissement de bulletins de paye
- fichier des records dans un club sportif
- etc...

```
        FILE STRUCTURE

      NAME:   ARTICLES
      DATE:   4/29/83
ITEMS/PAGE:   10
PAGES USED:   10
PAGES LEFT:   153
```

FILE STRUCTURE

	ITEM	TYPE	WIDTH	DEC
1	ARTICLE	CHAR	15	0
2	REFER	INT	10	0
3	IMPORT $	DEC	7	2
4	BASE $	DEC	7	2
5	BASE FF	DEC	8	2
6	CAT.1	DEC	8	2
7	CAT.2	DEC	8	2
8	CAT.3	DEC	8	2
9	CAT.4	DEC	8	2
10	COURS $	DEC	5	2

Tableau 8

```
     DISPLAY PAGE

PAGE #?      1

 1 ARTICLE    TELE
 2 REFER      20483
 3 IMPORT $   300.00
 4 BASE $
 5 BASE FF
 6 CAT.1
 7 CAT.2
 8 CAT.3
 9 CAT.4
10 COURS $    6.50
```

Tableau 9

```
     ITEM STATISTICS

    ITEM =   BASE FF
    MEAN =   1424.061
 STD DEV =   924.9194403
 MAX VAL =   2827.5
 MIN VAL =   85.96
     SUM =   14240.61
    DATA =   10
 MISSING =   0
```

Tableau 12

FILE: ARTICLES
DATE: 5/21/83
TITLE: CATALOGUE DE VENTE

0	1	2	6	7	8	9
1	TELE	20483	3675.75	3393.00	3251.63	3110.25
2	HIFI	10983	3375.56	3115.91	2986.08	2856.25
3	CALCUL.	92037	1075.16	992.45	951.10	909.75
4	DISQUES	32010	111.74	103.15	98.85	94.55
5	CALCUL.	92134	393.92	363.62	348.47	333.32
6	HIFI	32061	2209.13	2039.19	1954.23	1869.26
7	MAGNETOS	57182	2089.66	1928.92	1848.55	1768.18
8	MAGNETOS	57981	1228.31	1133.83	1086.58	1039.34
9	PLATINES	21645	1535.48	1417.37	1358.31	1299.26
10	PLATINES	21109	2818.08	2601.30	2492.91	2384.53

Tableau 11

```
FILE:  ARTICLES              INDEX             5 = BASE FF
DATE:  5/21/83               0 = PAGE #        6 = CAT.1
TITLE: CATALOGUE DE PRIX     1 = ARTICLE       7 = CAT.2
                             2 = REFER         8 = CAT.3
                             3 = IMPORT $      9 = CAT.4
                             4 = BASE $       10 = COURS $
```

0	1	2	3	4	5	6	7	8	9	10
1	TELE	20483	300.00	435.00	2827.50	3675.75	3393.00	3251.63	3110.25	6.50
2	HIFI	10983	275.50	399.48	2596.59	3375.56	3115.91	2986.08	2856.25	6.50
3	CALCUL.	92037	87.75	127.24	827.04	1075.16	992.45	951.10	909.75	6.50
4	DISQUES	32010	9.12	13.22	85.96	111.74	103.15	98.85	94.55	6.50
5	CALCUL.	92134	32.15	46.62	303.01	393.92	363.62	348.47	333.32	6.50
6	HIFI	32061	180.30	261.44	1699.33	2209.13	2039.19	1954.23	1869.26	6.50
7	MAGNETOS	57182	170.55	247.30	1607.43	2089.66	1928.92	1848.55	1768.18	6.50
8	MAGNETOS	57981	100.25	145.36	944.86	1228.31	1133.83	1086.58	1039.34	6.50
9	PLATINES	21645	125.32	181.71	1181.14	1535.48	1417.37	1358.31	1299.26	6.50
10	PLATINES	21109	230.00	333.50	2167.75	2818.08	2601.30	2492.91	2384.53	6.50

Tableau 10

Quand le TI devient un harmonium

Gilbert Arribet

L'instruction CALL KEY permet d'appeler un son à partir d'une touche du clavier. Nous l'avons utilisée pour transformer votre TI-99 en un instrument de musique; les notes, sur le clavier de la console, se présentent comme sur un piano :

W		E		T	Y	U	
A	S	D	F	G	H	J	K
do	ré	mi	fa	si	la	si	do

Le clavier de cet instrument comporte les notes d'une octave, plus la première note de l'octave suivante, ce qui n'est pas suffisant pour jouer la plupart des notes de musique. Aussi avons-nous ajouté la possibilité de changer d'octave : il est possible de choisir cinq octaves en appuyant sur les touches 1, 2, 3, 4 ou 5.

Le timbre de cet instrument s'apparente plus à celui d'un harmonium qu'à celui du piano, d'où le titre de cet article. Autant le dire tout de suite, au risque de décevoir certains : cet harmonium n'a aucunement la prétention d'égaler un véritable instrument de musique.

Pour le réaliser, nous allons procéder par étapes :

- la première partie, la plus importante, traite de la production des notes ;
- la seconde partie permet de "meubler" l'écran : le clavier de l'instrument est dessiné et un index précise l'octave choisie ;
- dans la troisième et dernière partie, le nom de l'instrument est affiché et un petit pavé clignotant indique la touche pressée.

Première partie : production des notes

Les féquences des 13 notes de base, N(1) à N(13), sont définies par la boucle FOR...NEXT des lignes 1010 à 1030.

La fréquence du LA(1) (situé deux octaves au-dessous du LA(3) de fréquence 440 Hz), égale à 110 Hz sert de référence. Les fréquences de deux notes distantes d'un demi-ton étant dans le rapport 2△(1/12), la fréquence de la première note (I=1), qui est à trois demi-tons (I+2 =1+2=3) au-dessus du LA(1), est donc 110⋆2△(3/12); celle de la note située à n demi-tons au-dessus du LA(1) (n=I+2) est 110⋆2△(n/12).

A la ligne 1070, les fréquences des notes de l'octave de rang J (1, 2, 3, 4 ou 5) sont obtenues en multipliant celles des notes de base : N(I) par 2△(J−1), soit par :
2△(1-1)=2△0=1 pour la première octave (J=1)
2△(2-1)=2△1=2 pour la seconde octave (J=2)
2△(3-1)=2△2=4 pour la troisième octave (J=3)
2△(4-1)=2△3=8 pour la quatrième octave (J=4)
2△(5-1)=2△4=16 pour la cinquième octave (J=5).

Les valeurs entières F1(I,J) des fréquences obtenues sont mises en mémoire dans un tableau à deux dimensions. La ligne 1080 donne F2(I,J), harmonique de fréquence double de F1(I,J); le son obtenu par le mélange des deux fréquences a un timbre plus agréable que celui obtenu avec F1(I,J) seule. L'instruction CALL SOUND de la ligne 1090 permet d'entendre les notes lors de leur mise en mémoire. L'accord de la ligne 1150 signale la fin de cette mise en mémoire.

Vous vous posez peut-être la question : pourquoi ne pas utiliser, pour les notes de base, les fréquences données par les tables (131 Hz, 139 Hz, ...247 Hz, 262 Hz), plutôt que la formule complexe de la ligne 1020 ? La raison est la suivante : les valeurs des fréquences données par les tables sont arrondies et, pour certaines d'entre elles, quand on les multiplie par 8 et surtout par 16 (octaves 4 et 5), l'écart avec la valeur précise de la fréquence est sensible à une oreille moyennement exercée, et l'instrument "joue faux" !

L'instruction CALL KEY de la ligne 1220 permet :

- de choisir l'octave (lignes 1770 à 1960). Exemple : si on appuie sur la touche "1" (code ASCII 49), l'octave 1 est sélectionnée...
- d'appeler une note en appuyant sur une touche déterminée (lignes 1240 à 1750). Exemple, si on appuye sur la touche "A" (code ASCII 65), la note DO (I=1) est jouée...

Le son de fréquence F1(I,J), avec son harmonique F2(I,J) est généré aux lignes 2280 ou 2320. Les instructions CALL SOUND des touches blanches et des touches noires sont séparées afin de permettre ultérieurement (troisième partie) l'indication de la touche pressée. Dans ces CALL SOUND, la durée négative permet d'obtenir un son "tenu" si l'appui sur la touche persiste; il faut régler la valeur absolue de cette durée à la plus faible valeur qui permette d'obtenir un son continu quand les touches les plus à droite du clavier (U ou K) sont maintenues enfoncées (cette valeur est ici de 140 millisecondes).

Une fois cette première partie du programme entrée, l'harmonium doit fonctionner ; mais, lors de son utilisation, on ne sait pas sur quelle touche appuyer pour obtenir une note donnée. Une première solution consisterait à coller des étiquettes, portant le nom des notes, sur les touches correspondantes (DO sur A...) Nous avons préféré dessiner sur l'écran le clavier avec l'indication des touches. C'est ce que nous ferons dans la seconde partie. Nous y réaliserons aussi une indication de l'octave choisie, ce qui est indispensable car, dans l'état actuel de l'instrument, rien ne permet de le savoir.

Seconde partie : dessin du clavier

Une boucle FOR...NEXT (lignes 630 à 670) trace les touches noires. L'instruction IF...THEN de la ligne 640 permet de ne pas afficher la troisième touche noire (intervalle d'un demi-ton entre MI et FA).

Le marquage des touches blanches (lignes 540 à 610) se fait sans problème. Au contraire, celui des touches noires, par les lettres W, E, T, Y et U, ne peut se faire simplement car ces lettres se trouvent à cheval sur deux caractères. Il a donc fallu redessiner chacune de ces lettres à l'aide de deux caractères (lignes 700 à 930).

L'indication d'octave se fait par le sous-programme défini aux lignes 2230 et 2240.

Troisième partie : identification de la touche

Elle se fait par un pavé clignotant (celui des touches noires étant défini par deux caractères). Conséquence de cette indication de la touche pressée : dans les CALL SOUND des lignes 2280 et 2320, la durée de 140 ms n'est plus suffisante pour obtenir un son continu quand on maintient

appuyées les touches U ou K ; il faut donc augmenter cette durée. Par exemple :

	Touches blanches	Touches noires
En TI-Basic sans le module Basic étendu	300 ms	340 ms
En TI-Basic avec le module	340 ms	370 ms
En Basic étendu	250 ms	250 ms

Le passage d'une note à une autre se fait donc plus lentement, cette augmentation de "l'inertie" de l'instrument peut justifier le non établissement ou la suppression de cette indication de la touche pressée.

Nom de l'instrument

Pour ne pas avoir les caractères A, H, I, M, N, O, R et U en noir sur fond blanc (couleur des groupes 3 à 10 définies aux lignes 170 à 190), leurs codes ASCII ont été redéfinis dans les groupes 1 et 2 non encore utilisés. Les couleurs de caractères de ces groupes sont définies aux lignes 2090 et 2100.

L'affichage des caractères peut se faire par une boucle FOR...NEXT ; remplacer à cet effet les lignes 2120 à 2300 par :

```
2120 DATA 2, 18, 40, 3, 19, 35, 3,
     21, 36, 3, 23, 37, 4, 20, 41
2125 DATA 4, 22, 42, 4, 24, 43, 5,
     25, 38, 6, 26, 44
2130 FOR K= 1 TO 9
2140 READ Y,X,CH
2150 CALL HCHAR(Y,X,CH)
2160 NEXT K
```

```
50 REM ====================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET L'AUTEUR
80 REM ====================
90 REM
100 REM  * HARMONIUM *
110 REM  PREMIERE PARTIE
120 REM  AUTEUR: G. ARRIBET
130 REM
150 CALL CLEAR
940 REM MISE EN MEMOIRE DES NOTES (5 OCTAVES
)
950 REM
960 REM "I": NUMEROS DES NOTES D'UNE OCTAVE;
 "J": NUMEROS DES OCTAVES
970 REM  "N(I)": FREQUENCES DES NOTES DE LA
PREMIERE OCTAVE
980 REM "F1(I,J)": FREQUENCE DE LA NOTE "I",
 DE L'OCTAVE "J"
990 REM "F2(I,J)": HARMONIQUE 2 DE "F1(I,J)"
1000 DIM N(13),F1(13,5),F2(13,5)
1010 FOR I=1 TO 13
1020 N(I)=110*(2^((I+2)/12))
1030 NEXT I
1040 FOR J=1 TO 5
1060 FOR I=1 TO 13
1070 F1(I,J)=INT(N(I)*(2^(J-1)))
1080 F2(I,J)=2*F1(I,J)
1090 CALL SOUND(50,F1(I,J),2,F2(I,J),6)
1100 NEXT I
1110 FOR T=1 TO 200
1120 NEXT T
1130 NEXT J
1140 REM FIN DE LA MISE EN MEMOIRE
1150 CALL SOUND(2000,262,10,330,10,392,10)
1160 J=1
1200 REM L'HARMONIUM EST PRET A JOUER
1210 REM  APPEL DE LA NOTE (TOUCHES A,W,S,E,
D,F,T,G,Y,H,U,J,K)
1220 CALL KEY(3,TH,ET)
1240 IF TH<>65 THEN 1280
1250 I=1
1270 GOTO 2280
1280 IF TH<>87 THEN 1320
1290 I=2
1310 GOTO 2320
1320 IF TH<>83 THEN 1360
1330 I=3
1350 GOTO 2280
1360 IF TH<>69 THEN 1400
1370 I=4
1390 GOTO 2320
1400 IF TH<>68 THEN 1440
1410 I=5
1430 GOTO 2280
1440 IF TH<>70 THEN 1480
1450 I=6
1470 GOTO 2280
1480 IF TH<>84 THEN 1520
1490 I=7
1510 GOTO 2320
1520 IF TH<>71 THEN 1560
1530 I=8
1550 GOTO 2280
1560 IF TH<>89 THEN 1600
1570 I=9
1590 GOTO 2320
1600 IF TH<>72 THEN 1640
1610 I=10
1630 GOTO 2280
1640 IF TH<>85 THEN 1680
1650 I=11
1670 GOTO 2320
1680 IF TH<>74 THEN 1720
1690 I=12
1710 GOTO 2280
1720 IF TH<>75 THEN 1770
1730 I=13
1750 GOTO 2280
1760 REM CHOIX DE L'OCTAVE (J=1,2,3,4,5)
1770 IF TH<>49 THEN 1810
1780 J=1
1800 GOTO 1220
1810 IF TH<>50 THEN 1850
1820 J=2
1840 GOTO 1220
1850 IF TH<>51 THEN 1890
1860 J=3
1880 GOTO 1220
1890 IF TH<>52 THEN 1930
1900 J=4
1920 GOTO 1220
1930 IF TH<>53 THEN 1220
1940 J=5
1960 GOTO 1220
2260 REM OBTENTION DU SON
2270 REM TOUCHES BLANCHES
2280 CALL SOUND(-140,F1(I,J),2,F2(I,J),6)
2300 GOTO 1220
2310 REM TOUCHES NOIRES
2320 CALL SOUND(-140,F1(I,J),2,F2(I,J),6)
2350 GOTO 1220
```

```
50 REM ====================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET L'AUTEUR
80 REM ====================
90 REM
100 REM  * HARMONIUM *
110 REM  SECONDE PARTIE
120 REM  AUTEUR: G. ARRIBET
130 REM
140 CALL SCREEN(8)
150 CALL CLEAR
160 REM COULEUR DES CARACTERES: OCTAVES,TOUC
HES BLANCHES ET NOIRES
170 FOR G=3 TO 10
180 CALL COLOR(G,2,16)
190 NEXT G
200 REM COULEUR DE L'INDEX D'OCTAVE (^)
210 CALL COLOR(11,9,8)
260 REM DEFINITION DE CARACTERES:
270 REM INDEX D'OCTAVE "^" (REDEFINITION DU
CODE ASCII DE "^")
280 CALL CHAR(112,"0000102844")
340 REM AFFICHAGE DE L'ECRAN (LIGNES 340 A 9
30)
350 PRINT "OCTAVES": : : : : : : : : : : : :
 : : : : : : : :
380 REM NUMEROS DES OCTAVES
390 FOR J=1 TO 5
400 CALL HCHAR(5,2+2*J,48+J)
410 NEXT J
420 REM TRACE DES TOUCHES BLANCHES
430 CALL CHAR(97,"")
```

```
440 CALL CHAR(98,"8080808080808080")
450 CALL CHAR(99,"FFFFFFFFFFFFFFFF")
460 FOR K=5 TO 27
470 CALL VCHAR(10,K,97,12)
480 NEXT K
490 CALL VCHAR(17,28,97,5)
500 FOR K=8 TO 26 STEP 3
510 CALL VCHAR(10,K,98,12)
520 NEXT K
530 REM MARQUAGE DES TOUCHES BLANCHES (A,S,D
,F,G,H,J,K)
540 CALL HCHAR(19,6,65)
550 CALL HCHAR(19,9,83)
560 CALL HCHAR(19,12,68)
570 CALL HCHAR(19,15,70)
580 CALL HCHAR(19,18,71)
590 CALL HCHAR(19,21,72)
600 CALL HCHAR(19,24,74)
610 CALL HCHAR(19,27,75)
620 REM TRACE DES TOUCHES NOIRES
630 FOR K=7 TO 22 STEP 3
640 IF K=13 THEN 670
650 CALL VCHAR(10,K,99,7)
660 CALL VCHAR(10,K+1,99,7)
670 NEXT K
680 REM MARQUAGE DES TOUCHES NOIRES (W,E,T,Y
,U)
690 REM CARACTERE "W"
700 CALL CHAR(100,"E0E4E4E5E5E5E2E")
710 CALL CHAR(101,"0F4F4F4F4F4F8F0F")
720 CALL HCHAR(14,7,100)
730 CALL HCHAR(14,8,101)
740 REM CARACTERE "E"
750 CALL CHAR(102,"E0E7E4E7E7E4E7E")
760 CALL CHAR(103,"0FCF0F0F0F0FCF0F")
770 CALL HCHAR(14,10,102)
780 CALL HCHAR(14,11,103)
790 REM CARACTERE "T"
800 CALL CHAR(104,"E0E7E1E1E1E1E1E")
810 CALL CHAR(105,"0FCF0F0F0F0F0F0F")
820 CALL HCHAR(14,16,104)
830 CALL HCHAR(14,17,105)
840 REM CARACTERE "Y"
850 CALL CHAR(106,"E0E4E4E2E1E1E1E")
860 CALL CHAR(107,"0F4F4F8F0F0F0F0F")
870 CALL HCHAR(14,19,106)
880 CALL HCHAR(14,20,107)
890 REM CARACTERE "U"
900 CALL CHAR(108,"E0E4E4E4E4E4E3E")
910 CALL CHAR(109,"0F4F4F4F4F4F8F0F")
920 CALL HCHAR(14,22,108)
930 CALL HCHAR(14,23,109)
1050 GOSUB 2230
1170 GOSUB 2230
1790 GOSUB 2230
1830 GOSUB 2230
1870 GOSUB 2230
1910 GOSUB 2230
1950 GOSUB 2230
2220 REM INDICATION DE L'OCTAVE: POSITION DE
 L'INDEX (^)
2230 CALL HCHAR(6,2,32,12)
2240 CALL HCHAR(6,2+2*J,112)
2250 RETURN
```

```
50 REM =====================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET L'AUTEUR
80 REM =====================
90 REM
100 REM  * HARMONIUM *
110 REM  TROISIEME PARTIE
120 REM  AUTEUR: G. ARRIBET
130 REM
220 REM COULEUR DU PAVE CLIGNOTANT SUR LES T
OUCHES NOIRES
230 CALL COLOR(12,4,2)
240 REM COULEUR DU PAVE CLIGNOTANT SUR LES T
OUCHES BLANCHES
250 CALL COLOR(13,9,16)
290 REM PAVE CLIGNOTANT SUR LES TOUCHES NOIR
ES
300 CALL CHAR(120,"0000000003030303")
310 CALL CHAR(121,"00000000C0C0C0C")
320 REM PAVE CLIGNOTANT SUR LES TOUCHES BLAN
CHES
330 CALL CHAR(128,"000038383838")
360 REM NOM DE L'INSTRUMENT: "HARMONIUM"
370 GOTO 1990
1180 X1=6
1190 X2=7
1230 GOTO 2370
1260 X1=6
1300 X2=7
1340 X1=9
1380 X2=10
1420 X1=12
1460 X1=15
1500 X2=16
1540 X1=18
1580 X2=19
1620 X1=21
1660 X2=22
1700 X1=24
1740 X1=27
1970 REM AFFICHAGE DU NOM DE L'INSTRUMENT: "
HARMONIUM"
1980 REM REDEFINITION DES CODES ASCII DES CA
RACTERES A,H,I,M,N,O,R,U
1990 CALL CHAR(35,"003844447C444444")
2000 CALL CHAR(36,"00446C5454444444")
2010 CALL CHAR(37,"00446464544C4C44")
2020 CALL CHAR(38,"0044444444444438")
2030 CALL CHAR(40,"004444447C444444")
2040 CALL CHAR(41,"0078444478504844")
2050 CALL CHAR(42,"007C44444444447C")
2060 CALL CHAR(43,"0038101010101038")
2070 CALL CHAR(44,"00446C5454444444")
2080 REM COULEUR DES CARACTERES
2090 CALL COLOR(1,9,8)
2100 CALL COLOR(2,13,8)
2110 REM AFFICHAGE DES CARACTERES
2120 CALL HCHAR(2,18,40)
2130 CALL HCHAR(3,19,35)
2140 CALL HCHAR(3,21,36)
2150 CALL HCHAR(3,23,37)
2160 CALL HCHAR(4,20,41)
2170 CALL HCHAR(4,22,42)
2180 CALL HCHAR(4,24,43)
2190 CALL HCHAR(5,25,38)
2200 CALL HCHAR(6,26,44)
2210 GOTO 390
2260 REM OBTENTION DU SON ET INDICATION DE L
A TOUCHE PRESSEE (PAVES):
2270 REM TOUCHES BLANCHES
2290 CALL HCHAR(20,X1,128)
2310 REM TOUCHES NOIRES
2330 CALL HCHAR(15,X2,120)
2340 CALL HCHAR(15,X2+1,121)
2360 REM EFFACEMENT DES PAVES SUR LES TOUCHE
S BLANCHES ET NOIRES
2370 CALL HCHAR(20,X1,97)
2380 CALL HCHAR(15,X2,99,2)
2390 GOTO 1240
```

```
50 REM =====================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET L'AUTEUR
80 REM =====================
90 REM
100 REM  * HARMONIUM *
110 REM
120 REM  AUTEUR: G. ARRIBET
130 REM
140 CALL SCREEN(8)
150 CALL CLEAR
160 REM COULEUR DES CARACTERES: OCTAVES,TOUC
HES BLANCHES ET NOIRES
170 FOR G=3 TO 10
180 CALL COLOR(G,2,16)
190 NEXT G
200 REM COULEUR DE L'INDEX D'OCTAVE (^)
210 CALL COLOR(11,9,8)
220 REM COULEUR DU PAVE CLIGNOTANT SUR LES T
OUCHES NOIRES
230 CALL COLOR(12,4,2)
240 REM COULEUR DU PAVE CLIGNOTANT SUR LES T
OUCHES BLANCHES
250 CALL COLOR(13,9,16)
260 REM DEFINITION DE CARACTERES:
270 REM INDEX D'OCTAVE "^" (REDEFINITION DU
CODE ASCII DE "^")
280 CALL CHAR(112,"0000102844")
290 REM PAVE CLIGNOTANT SUR LES TOUCHES NOIR
ES
300 CALL CHAR(120,"0000000003030303")
310 CALL CHAR(121,"00000000C0C0C0C")
320 REM PAVE CLIGNOTANT SUR LES TOUCHES BLAN
CHES
```

```
330 CALL CHAR(128,"000038383838")
340 REM AFFICHAGE DE L'ECRAN (LIGNES 340 A 9
30)
350 PRINT "OCTAVES": : : : : : : : : : : : :
 : : : : : : : :
360 REM NOM DE L'INSTRUMENT: "HARMONIUM"
370 GOTO 1990
380 REM NUMEROS DES OCTAVES
390 FOR J=1 TO 5
400 CALL HCHAR(5,2+2*J,48+J)
410 NEXT J
420 REM TRACE DES TOUCHES BLANCHES
430 CALL CHAR(97,"")
440 CALL CHAR(98,"8080808080808080")
450 CALL CHAR(99,"FFFFFFFFFFFFFFFF")
460 FOR K=5 TO 27
470 CALL VCHAR(10,K,97,12)
480 NEXT K
490 CALL VCHAR(17,28,97,5)
500 FOR K=8 TO 26 STEP 3
510 CALL VCHAR(10,K,98,12)
520 NEXT K
530 REM MARQUAGE DES TOUCHES BLANCHES (A,S,D
,F,G,H,J,K)
540 CALL HCHAR(19,6,65)
550 CALL HCHAR(19,9,83)
560 CALL HCHAR(19,12,68)
570 CALL HCHAR(19,15,70)
580 CALL HCHAR(19,18,71)
590 CALL HCHAR(19,21,72)
600 CALL HCHAR(19,24,74)
610 CALL HCHAR(19,27,75)
620 REM TRACE DES TOUCHES NOIRES
630 FOR K=7 TO 22 STEP 3
640 IF K=13 THEN 670
650 CALL VCHAR(10,K,99,7)
660 CALL VCHAR(10,K+1,99,7)
670 NEXT K
680 REM MARQUAGE DES TOUCHES NOIRES (W,E,T,Y
,U)
690 REM CARACTERE "W"
700 CALL CHAR(100,"E0E4E4E5E5E5E2E")
710 CALL CHAR(101,"0F4F4F4F4F4F8F0F")
720 CALL HCHAR(14,7,100)
730 CALL HCHAR(14,8,101)
740 REM CARACTERE "E"
750 CALL CHAR(102,"E0E7E4E7E7E4E7E")
760 CALL CHAR(103,"0FCF0F0F0F0FCF0F")
770 CALL HCHAR(14,10,102)
780 CALL HCHAR(14,11,103)
790 REM CARACTERE "T"
800 CALL CHAR(104,"E0E7E1E1E1E1E1E")
810 CALL CHAR(105,"0FCF0F0F0F0F0F0F")
820 CALL HCHAR(14,16,104)
830 CALL HCHAR(14,17,105)
840 REM CARACTERE "Y"
850 CALL CHAR(106,"E0E4E4E2E1E1E1E")
860 CALL CHAR(107,"0F4F4F8F0F0F0F0F")
870 CALL HCHAR(14,19,106)
880 CALL HCHAR(14,20,107)
890 REM CARACTERE "U"
900 CALL CHAR(108,"E0E4E4E4E4E4E3E")
910 CALL CHAR(109,"0F4F4F4F4F4F8F0F")
920 CALL HCHAR(14,22,108)
930 CALL HCHAR(14,23,109)
940 REM MISE EN MEMOIRE DES NOTES (5 OCTAVES
)
950 REM
960 REM "I": NUMEROS DES NOTES D'UNE OCTAVE;
 "J": NUMEROS DES OCTAVES
970 REM  "N(I)": FREQUENCES DES NOTES DE LA
PREMIERE OCTAVE
980 REM "F1(I,J)": FREQUENCE DE LA NOTE "I",
 DE L'OCTAVE "J"
990 REM "F2(I,J)": HARMONIQUE 2 DE "F1(I,J)"
1000 DIM N(13),F1(13,5),F2(13,5)
1010 FOR I=1 TO 13
1020 N(I)=110*(2^((I+2)/12))
1030 NEXT I
1040 FOR J=1 TO 5
1050 GOSUB 2230
1060 FOR I=1 TO 13
1070 F1(I,J)=INT(N(I)*(2^(J-1)))
1080 F2(I,J)=2*F1(I,J)
1090 CALL SOUND(150,F1(I,J),2,F2(I,J),6)
1100 NEXT I
1110 FOR T=1 TO 200
1120 NEXT T
1130 NEXT J
1140 REM FIN DE LA MISE EN MEMOIRE
1150 CALL SOUND(2000,262,10,330,10,392,10)
1160 J=1
1170 GOSUB 2230
1180 X1=6
1190 X2=7
1200 REM L'HARMONIUM EST PRET A JOUER
1210 REM  APPEL DE LA NOTE (TOUCHES A,W,S,E,
D,F,T,G,Y,H,U,J,K)
1220 CALL KEY(3,TH,ET)
1230 GOTO 2370
1240 IF TH<>65 THEN 1280
1250 I=1
1260 X1=6
1270 GOTO 2280
1280 IF TH<>87 THEN 1320
1290 I=2
1300 X2=7
1310 GOTO 2320
1320 IF TH<>83 THEN 1360
1330 I=3
1340 X1=9
1350 GOTO 2280
1360 IF TH<>69 THEN 1400
1370 I=4
1380 X2=10
1390 GOTO 2320
1400 IF TH<>68 THEN 1440
1410 I=5
1420 X1=12
1430 GOTO 2280
1440 IF TH<>70 THEN 1480
1450 I=6
1460 X1=15
1470 GOTO 2280
1480 IF TH<>84 THEN 1520
1490 I=7
1500 X2=16
1510 GOTO 2320
1520 IF TH<>71 THEN 1560
1530 I=8
1540 X1=18
1550 GOTO 2280
1560 IF TH<>89 THEN 1600
1570 I=9
1580 X2=19
1590 GOTO 2320
1600 IF TH<>72 THEN 1640
1610 I=10
1620 X1=21
1630 GOTO 2280
1640 IF TH<>85 THEN 1680
1650 I=11
1660 X2=22
1670 GOTO 2320
1680 IF TH<>74 THEN 1720
1690 I=12
1700 X1=24
1710 GOTO 2280
1720 IF TH<>75 THEN 1770
1730 I=13
1740 X1=27
1750 GOTO 2280
1760 REM CHOIX DE L'OCTAVE (J=1,2,3,4,5)
1770 IF TH<>49 THEN 1810
1780 J=1
1790 GOSUB 2230
1800 GOTO 1220
1810 IF TH<>50 THEN 1850
1820 J=2
1830 GOSUB 2230
1840 GOTO 1220
1850 IF TH<>51 THEN 1890
1860 J=3
1870 GOSUB 2230
1880 GOTO 1220
1890 IF TH<>52 THEN 1930
1900 J=4
1910 GOSUB 2230
1920 GOTO 1220
1930 IF TH<>53 THEN 1220
1940 J=5
1950 GOSUB 2230
1960 GOTO 1220
1970 REM AFFICHAGE DU NOM DE L'INSTRUMENT: "
HARMONIUM"
```

Suite page 65

ATTENTION :
seul le programme 4 doit être tapé. Les programmes 1, 2 et 3 sont là à titre d'illustration.

Fichiers en BASIC

Simon Tchek

Le présent article a pour but d'amener le lecteur qui dispose de la configuration minimale suivante :
- console TI-99/4A
- une unité de disquette

à créer des fichiers de données, autrement qu'en utilisant les modules ou programmes commercialisés à cet effet par Texas-Instruments, tels que "Gestion Personnelle de Fichiers", analysé dans un autre article de "99". Notre approche est cependant différente. Nous allons successivement :
- rechercher une meilleure compréhension du Basic
- poser des problèmes de gestion de l'information auxquels nous apporterons les solutions que nous pensons les plus simples.

Ces dernières années, le coût du matériel a constamment baissé ; de la sorte, de plus en plus de personnes peuvent disposer de configurations relativement importantes pour des utilisations familiales ou pour de petites applications professionnelles. Le prix des programmes n'a cependant pas baissé dans des proportions identiques. Dans ces conditions, presque tous les utilisateurs individuels sont amenés à programmer eux-mêmes leurs applications. Ils s'aperçoivent alors qu'une marge importante existe entre le programmeur amateur et l'auteur de logiciels professionnels, marge que seule une information encore trop rare peut combler. Les manuels de base des ordinateurs offrent une profusion de détails sur les rudiments du Basic et ne s'occupent que peu des techniques nécessaires à l'utilisation rationnelle de disques.

Nous nous limiterons dans ce qui suit à l'exposé des techniques concernant les fichiers à accès direct qui constituent le principal avantage de la souplesse d'utilisation des fichiers sur disques. Les fichiers séquentiels conçus pour être stockés sur cassettes présentent des restrictions importantes et ne seront pas développés ici.

Caractéristiques des disques

Structure du disque

Les disques disponibles sur le marché varient par la taille :
- les disques durs dont on distingue deux catégories : disques fixes et disques amovibles ;
- les disques souples, appelés "disquettes", dont la lecture se fait, contrairement aux disques durs, par contact direct de la tête magnétique avec la surface du disque, ce qui les rend sensiblement plus fragiles. Deux tailles de disques souples existent actuellement : 8 pouces et 5 pouces (en ignorant les fractions de pouces...).

Capacité du disque

Le TI-99/4A utilise des disques souples de 5 pouces 1/4, simple face et simple densité. Ces disques sont divisés en 40 pistes concentriques, chaque piste étant à son tour divisée en 9 secteurs. Un secteur contient 256 octets d'informations (un octet est formé de huit bits, c'est-à-dire huit chiffres binaires 0 ou 1).

La quantité d'informations que peut stocker un disque est donc approximativement de 90K (1K = 1024 octets. Si l'on utilise un TI-99/4A en configuration maximale (48K de mémoire et deux unités de disques), on dispose en fait de 138K "en ligne", soit une configuration tout à fait performante. Rappelons cependant que les disques durs stockent en général cinq millions d'octets (ou 10, ou 20,...) !

Système de gestion de disques (DOS : Disk Operating System)

Le DOS est un ensemble d'instructions qui agit tantôt comme un langage interprété, tantôt comme complément au Basic. Ses fonctions essentielles permettent d'initialiser un disque, de copier des disques entre eux ou d'établir le catalogue d'un disque, c'est-à-dire la liste des fichiers qui y sont contenus avec indication de leur nature.

Sans entrer dans des détails techniques, nous pouvons signaler que le DOS dispose un répertoire au début du disque. Lorsqu'un fichier doit être stocké sur le disque, le DOS met ce répertoire à jour en y incorporant le nom, la taille et le type du fichier ainsi que l'adresse du début de la zone du disque dans laquelle il sera stocké.

L'initialisation d'un disque est une opération indispensable : elle consiste à formater le disque en pistes et secteurs compatibles avec le lecteur utilisé. Un fichier est stocké sur plusieurs secteurs, chaque secteur contenant comme dernière information l'adresse du secteur suivant pour le même fichier. Il est ainsi possible d'effacer un fichier ancien et d'utiliser la place ainsi libérée pour y stocker un nouveau fichier, même si celui-ci est plus grand que l'ancien. Le DOS s'occupe de l'ensemble de ces opérations.

Concepts de base d'un fichier

Le problème consiste à définir un objet à partir d'un groupe d'informations en vue de l'accomplissement d'une tâche. Ce groupe d'informations constitue les "champs" de chaque enregistrement du fichier (on peut dire aussi les rubriques, les items, les articles, etc...). Chaque fichier qui sera constitué aura sa propre structure, c'est-à-dire que chacun des enregistrements d'un fichier donné comporte toujours le même nombre de rubriques.

Par exemple, un fichier constitué en vue de la recherche d'un logement va repérer les logements visités par rapport au nombre de pièces, nombre de salles de bains, superficie en mètres carrés, quartier, montant du loyer et des charges.

En tant que locataire, l'agence à laquelle vous avez confié cette recherche s'intéresse à votre personne par rapport aux données : nom, prénom, âge au moment de la recherche, profession, revenu, etc...

Un professeur pourra repérer ses élèves par leurs nom, prénom, note 1, note 2,..., moyenne, etc...

Dernier exemple enfin : le classement des livres de votre biliothèque personnelle : titre, auteur, date de publication, date d'acquisition, éditeur, nombre de pages, catégorie (roman, documentation scientifique, etc...), prix d'achat, langue (français, anglais, etc...)

Ceci nous conduit à la définition suivante : Un *enregistrement* est un groupe d'informations caractérisant une entité.

Chaque entité du fichier est en effet composée d'une série d'informa-

tions : certaines caractérisent le logement, la personne, l'élève ou le livre. L'enregistrement peut alors être divisé en sous-groupes irréductibles, d'où une nouvelle définition : le *champ* (rubrique, item, article, élément...) est une partie irréductible d'un enregistrement caractérisant une entité.
L'enregistrement Logement, par exemple, comporte 6 champs; l'enregistrement Personne comporte 5 champs, etc..

Plusieurs enregistrements peuvent comporter des champs identiques. Par exemple, deux appartements peuvent être situés dans le même quartier. Certains champs, en revanche, suffisent à caractériser un enregistrement par rapport à tous les autres enregistrements du même fichier.
On repérera par exemple un livre dans la bibliothèque en l'identifiant par son titre : il n'y a pas deux livres qui ont le même titre si l'on considère comme une seule unité un livre en plusieurs tomes. En revanche, on peut posséder plusieurs livres du même auteur.

Types de champs

Un fichier peut comporter plusieurs types de champs :
- les champs numériques contiennent des nombres qui pourront faire l'objet de calculs ;
- les champs alphanumériques comportent toutes sortes de caractères. Si des nombres y sont inscrits, il ne pourront pas être directement utilisés pour des calculs.

Les champs numériques peuvent aussi être distingués les uns des autres selon qu'ils contiennent des nombres entiers, des nombres décimaux, des nombres en notation scientifique ou des dates.
Les champs alphanumériques, quant à eux, sont définis par leur longueur maximale autorisée.

Deux types essentiels d'organisation des fichiers peuvent être utilisés :

- les fichiers à accès séquentiel : tous les enregistrements y sont rangés dans un ordre déterminé. C'est par exemple le cas de tous les fichiers rangés sur bandes magnétiques;
- les fichiers dits à accès aléatoire, ce qui ne signifie pas que l'on accède aux informations par hasard, mais seulement qu'elles y sont rangées sans ordre défini. Cela est très souvent le cas des fichiers rangés sur disques où un index permet d'accéder directement à l'information recherchée. Nous parlerons donc plutôt de fichiers à accès direct.

les clés

Cela nous conduit à définir le concept de clé : une *clé* est un contenu de champ permettant de retrouver dans un fichier donné un ou plusieurs enregistrements comportant un champ de contenu identique. Ainsi :
- une clé primaire permet de sélectionner rapidement un enregistrement; en général, une clé primaire effectue cette sélection sans ambiguïté;
- une clé secondaire permet de sélectionner un enregistrement plus lentement, en général parce qu'elle réalise la recherche sur le support magnétique.

Un bon exemple de clé primaire est celui du numéro d'enregistrement : chaque enregistrement dans un fichier comporte un numéro différent de celui des autres enregistrements (si toutefois on a prévu que les enregistrements soient numérotés). Une clé secondaire peut être par exemple le nom d'un auteur dans un fichier de livres : ces noms seront probablement sur la disquette et non résidents en mémoire.

Le programme ci-contre réalise un embryon de gestion de fichier : il s'agit là de constituer un fichier comportant les noms des élèves d'une classe et la note qu'ils ont obtenue sur un devoir donné. Le menu de départ conduit vers 6 tâches distinctes. Le fichier constitué par ce programme est un fichier à accès direct : les enregistrements y sont rangés sans ordre défini. Vous constatez à la ligne 1060 que chaque enregistrement est numéroté, ce qui permettra de le retrouver par la suite (interrogation par numéro, lignes 3000 à 3080). La numérotation choisie ici progresse de 1 en 1 à chaque nouvel enregistrement. On aurait pu, éventuellement, faire quelque chose de plus compliqué.

Une fois que le fichier est constitué, il est possible d'en obtenir des informations globales : moyenne des notes des élèves, écart-type, détermination de la note la plus haute ou la plus basse, etc. Nous avons même prévu un module dans ce programme permettant de faire un histogramme. Nous obtenons ainsi une représentation graphique des valeurs numériques insérées dans les enregistrements du fichier (lignes 8000 à 10070).

Conclusion

Il ne s'agit ici évidemment que d'un programme de démonstration qui réalise la gestion d'un fichier particulièrement simple. Cependant, il illustre à notre avis très bien ce que peut être une gestion de fichier. La constitution d'un fichier plus complexe ne revient finalement qu'à ajouter des rubriques à chaque enregistrement. En revanche, des différences importantes peuvent être apportées dans l'exploitation du fichier. On peut en effet prévoir des tâches plus compliquées qu'un simple calcul de moyenne et d'écart-type. C'est en fait cela qui est le plus ardu dans les programmes de gestion de fichiers.

```
50 REM ====================
60 REM COPYRIGHT 99 MAGAZINE
70 REM ET SIMON TCHEK
80 REM ====================
90 REM
100 REM =====FICHIER DE NOTES=====
110 REM ===MODULE PRINCIPAL
112 REM  ===ACCES DIRECT
113 REM  ==300 FICHES(1-300)
114 REM   ===FICHE:1RUBRIQUE         AL
PHA+1 RUBRIQUE           NUMERIQUE
120 CALL CLEAR
130 F$="NOTES"
140 OPEN #2:"DSK1."&F$,RELATIVE 301,INTE
RNAL
150 CALL CLEAR
160 PRINT "GESTION DU FICHIER:"&F$
170 PRINT : :"1=CREATION"
190 PRINT :"2=LECTURE DU FICHIER"
200 PRINT :"3=INTERROGATION(PAR NUMERO):
"
210 PRINT :"4=MODIFICATION D'UNE FICHE:"
220 PRINT :"5=AJOUT D'UNE FICHE:"
230 PRINT :"6=SORTIE DU PROGRAMME:"
235 PRINT :"7=EXPLOITATION:"
237 PRINT
240 INPUT "VOTRE CHOIX?:":X
250 PRINT
```

```
260 IF X<1 THEN 240
270 IF X>7 THEN 240
280 ON X GOSUB 1000,2000,3000,4000,5000,
6000,7000
320 GOTO 150
1000 REM ===MODULE DE CREATION
1010 PRINT "----------------------------"
1020 PRINT "ENTREE DES DONNEES":"-------
-------------------"
1030 PRINT "POUR SORTIR DU MODULE ENTREZ
 XXX,0"
1040 PRINT "----------------------------"
1050 TOTAL=0
1060 INPUT "FICHE("&STR$(TOTAL+1)&") ":B
UFFER$,NOTE
1070 IF BUFFER$="XXX" THEN 1130
1080 PRINT #2,REC TOTAL+1:BUFFER$,NOTE
1090 TOTAL=TOTAL+1
1100 IF TOTAL<300 THEN 1060
1110 REM  ===MESSAGE FICHIER PLEIN
1120 PRINT "FICHIER PLEIN"
1130 PRINT #2,REC 0:TOTAL
1140 RETURN
2000 REM  ===MODULE DE LECTURE
2002 PRINT "LECTURE PAR TRANCHES DE COMB
IEN DE FICHES?"
2004 INPUT G
2005 IF G<>INT(G)THEN 2002
2010 CALL CLEAR
2020 PRINT "LECTURE DU FICHIER:"&F$: :
2030 RESTORE #2,REC 0
2040 INPUT #2,REC 0:TOTAL
2050 PRINT "FICHES UTILISEES ";TOTAL:"LI
BRES:";300-TOTAL
2055 PRINT
2060 FOR COMPTEUR=1 TO TOTAL
2070 INPUT #2:BUFFER$,NOTE
2080 PRINT "FICHE("&STR$(COMPTEUR)&") ";
BUFFER$;NOTE
2085 W=COMPTEUR/G
2086 IF W=INT(W)THEN 2087 ELSE 2090
2087 GOSUB 2200
2090 NEXT COMPTEUR
2092 PRINT ::
2095 INPUT "POUR CONTINUER? (O/N)":A$
2096 IF A$="O" THEN 2100
2097 IF A$="N" THEN 6000
2100 RETURN
2200 PRINT "SUITE?(O/N)"
2210 INPUT R$
2220 IF R$="O" THEN RETURN
2230 IF R$="N" THEN 150
2240 GOTO 2200
3000 REM  INTERROGATION(PARNUMERO)
3010 CALL CLEAR
3020 PRINT "INTERROGATION(PAR NUMERO)":"
==========================": :
3030 GOSUB 10000
3040 INPUT #2,REC NUMFICHE:BUFFER$,NOTE
3045 PRINT
3050 PRINT "FICHE(";NUMFICHE;")";BUFFER$
;NOTE
3055 PRINT :"----------------------------
"
3060 INPUT "VOULEZ-VOUS D'AUTRES FICHES
                   (O/N)?":R$
3065 PRINT "----------------------------"
3070 IF R$="O" THEN 3030
3080 RETURN
4000 REM ===MODIFICATION D'UNE FICHE
4010 CALL CLEAR
4020 PRINT : :"MODIFICATION D'UNE FICHE"
:"----------------------------": :
4030 GOSUB 10000
4040 INPUT #2,REC NUMFICHE:BUFFER$,NOTE
4050 PRINT :"FICHE(";NUMFICHE;") ";BUFFE
R$;NOTE:"----------------------------"
4060 INPUT "NOUVELLES DONNEES:":BUFFER$,
NOTE
4070 PRINT #2,REC NUMFICHE:BUFFER$,NOTE
4075 PRINT :"----------------------------
";
4080 INPUT "Y A-T-IL D'AUTRES
MODIFICATIONS?(O/N):":M$
4085 PRINT :"----------------------------
":
4090 IF M$="O" THEN 4000
4100 RETURN
5000 REM ===AJOUT D'UNE FICHE
5010 CALL CLEAR
5020 PRINT "AJOUT D'UNE FICHE": :"-------
---------------------":
5030 INPUT #2,REC 0:TOTAL
5040 PRINT TAB(25);F$
5050 PRINT "FICHES"
5060 PRINT "DISPONIBLES";300-TOTAL
5070 PRINT "OCCUPEES:";TOTAL
5080 PRINT "----------------------------"
: :
5090 IF TOTAL>=300 THEN 5220
5092 INPUT #2,REC TOTAL:BUFFER$,NOTE
5093 PRINT "DERNIERE FICHE"&"("&STR$(TOT
AL)&")";BUFFER$;NOTE
5100 INPUT "ENTREZ VOS DONNEES:":BUFFER$
,NOTE
5110 PRINT #2,REC TOTAL+1:BUFFER$,NOTE
5120 PRINT #2,REC 0:TOTAL+1
5125 PRINT :"----------------------------
"
5130 INPUT "D'AUTRES AJOUTS?(O/N)":Q$
5200 TOTAL=TOTAL+1
5205 IF TOTAL>=301 THEN 5220
5207 IF Q$<>"O" THEN 5230
5210 GOTO 5060
5220 PRINT "FICHIER PLEIN"
5230 RETURN
6000 REM  ===MODULE DE SORTIE
6005 CALL CLEAR
```

```
6010 CLOSE #2
6020 PRINT "TOUTES LES DONNEES": :"SONT
ENREGISTREES": :
6030 END
7000 REM EXPLOITATION DU FICHIER NOTES==
========================================
====
7005 CALL CLEAR
7010 PRINT "1=MOYENNE ET ECART TYPE"
7020 PRINT
7030 PRINT "2=VALEURS EXTREMES"
7040 PRINT
7050 PRINT "3=HISTOGRAMME"
7060 PRINT
7065 PRINT "4=GESTION DU FICHIER"
7067 PRINT
7070 PRINT "VOTRE CHOIX?"
7080 INPUT Y
7082 IF Y=4 THEN 150
7085 DIM A(300)
7090 INPUT #2,REC 0:TOTAL
7100 FOR I=1 TO TOTAL
7110 INPUT #2:D$,A(I)
7120 NEXT I
7125 CALL CLEAR
7130 ON Y GOTO 7500,8000,8500
7500 REM CALCUL DE LA MOYENNE ET DE L'EC
ART-TYPE================================
====
7510 REM S=SOMME DES A(I),T=SOMME DES CA
RRES DES A(I)
7520 S=0
7530 T=0
7540 FOR I=1 TO TOTAL
7550 S=S+A(I)
7560 T=T+A(I)^2
7570 NEXT I
7580 M=S/TOTAL
7590 PRINT "MOYENNE=";M
7600 E=SQR(T*TOTAL-S^2)/TOTAL
7610 PRINT
7620 PRINT
7630 PRINT "ECART-TYPE=";E
7632 PRINT
7634 PRINT
7640 PRINT "G=GESTION DU FICHIER"
7650 PRINT
7670 PRINT "E=EXPLOITATION"
7680 PRINT
7685 PRINT "S=SORTIE"
7688 PRINT
7690 PRINT "VOTRE CHOIX?"
7700 INPUT Z$
7710 IF Z$="G" THEN 150
7720 IF Z$="E" THEN 7000
7730 IF Z$="S" THEN 6000
7999 GOTO 150
8000 REM CALCUL DES VALEURS EXTREMES====
========================================
====
8010 SUP=-999999999
8020 INF=999999999999
8030 FOR I=1 TO TOTAL
8040 IF A(I)<SUP THEN 8060
8050 SUP=A(I)
8060 IF A(I)>INF THEN 8080
8070 INF=A(I)
8080 NEXT I
8090 CALL CLEAR
8100 PRINT "MINIMUM=";INF
8110 PRINT
8120 PRINT "MAXIMUM=";SUP
8130 GOTO 7632
8499 GOTO 7000
8500 REM HISTOGRAMME====================
================
8502 DIM B(21)
8503 FOR I=0 TO 20
8504 B(I)=0
8505 NEXT I
8510 CALL CLEAR
8520 PRINT "VOS NOTES SONT SUR?"
8530 INPUT N
8532 FOR I=5 TO 7 :: CALL COLOR(I,16,1):
: NEXT I
8540 FOR I=1 TO TOTAL
8550 X=INT(A(I)/N*20)
8560 B(X)=B(X)+1
8570 NEXT I
8580 SUP=-1
8590 INF=10000000000000
8600 FOR X=0 TO 21
8610 IF B(X)<SUP THEN 8630
8620 SUP=B(X)
8630 IF B(X)>INF THEN 8650
8640 INF=B(X)
8650 NEXT X
8652 CALL CLEAR
8655 CALL CHAR(34,"0301010101010103")
8656 CALL VCHAR(1,9,34,22)
8660 CALL CHAR(33,"FF8181")
8665 CALL HCHAR(23,10,33,20)
8670 CALL HCHAR(23,10,33,20)
8680 CALL HCHAR(23,9,48)
8690 CALL HCHAR(23,30,50)
8700 CALL HCHAR(23,31,48)
8710 FOR J=9 TO 14
8720 CALL COLOR(J,9,1)
8730 NEXT J
8735 CALL CHAR(96,"FFFFFFFFFFFFFFFF")
8740 U=21/SUP
8742 C=97
8750 FOR X=0 TO 20
8760 H=B(X)*U
8770 REM HH=NOMBRE DE CARREAUX ENTIERS
8780 HH=INT(H)
8790 D=INT((H-HH)*8+0.5)
8800 IF HH<=0 THEN 8825
```

```
8810 CALL VCHAR(23-HH,X+10,96,HH)
8825 IF D=0 THEN 8870
8830 L$=SEG$("0000000000000000",1,16-2*D
)&SEG$("FFFFFFFFFFFFFFFF",1,2*D)
8840 CALL CHAR(C,L$)
8850 CALL VCHAR(22-HH,X+10,C)
8860 C=C+1
8870 NEXT X
8900 EU=1/U
8902 S=0 :: T=0 :: FOR I=1 TO TOTAL :: S
=S+A(I):: T=T+A(I)^2 :: NEXT I
8904 M=S/TOTAL :: MC=M*20/N
8905 E=SQR(T*TOTAL-S^2)/TOTAL :: EC=E*20
/N
8906 DISPLAY AT(10,1)SIZE(2):"M=" :: DIS
PLAY AT(10,3)SIZE(4):USING "##.#":MC
8908 DISPLAY AT(12,1)SIZE(2):"E=" :: DIS
PLAY AT(12,3)SIZE(4):USING "##.#":EC
8910 DISPLAY AT(22,1)SIZE(6):USING "###.
##":EU
8920 DISPLAY AT(2,1)SIZE(6):USING "###.#
#":SUP
8930 CALL CHAR(140,"0103050901010101000C
0A0908080800080C0A0908080808000305090101O
1000")
8940 CALL MAGNIFY(3)
8950 CALL SPRITE(#1,140,16,179,66+8*MC)
8960 FOR I=1 TO 20 :: NEXT I
8970 CALL COLOR(#1,1)
8980 FOR I=1 TO 20 :: NEXT I
9000 DISPLAY AT(24,1)SIZE(6):"SUITE?"
9005 CALL KEY(0,C,E)
9007 IF E<>0 THEN 9900
9010 GOTO 8950
9900 CALL CLEAR :: CALL DELSPRITE(ALL)::
 FOR I=5 TO 7 :: CALL COLOR(I,2,1):: NEX
T I
9905 GOTO 7632
9999 GOTO 9999
10000 INPUT "QUELLE FICHE?":NUMFICHE
10010 IF NUMFICHE<1 THEN 10050
10020 INPUT #2,REC 0:TOTAL
10030 IF NUMFICHE<=TOTAL THEN 10070
10050 PRINT "NUMERO INEXISTANT"
10060 GOTO 10000
10070 RETURN
```

Catalogue parlé sur les cassettes

On entend très bien, avec le haut-parleur du téléviseur, défiler tout ce qu'il y a sur les cassettes. Par contre, sauf à posséder le catalogue précis du contenu, il est impossible de savoir quel programme vous êtes en train de charger. Et ce catalogue a une fâcheuse tendance à disparaître...

Pour se prémunir contre cela, faites précéder vos programmes ou fichiers sur cassette d'une annonce sonore du genre de : "Programme du jeu de l'alphabet". Il vous faut positionner la bande à l'endroit où va commencer l'enregistrement, puis débrancher la fiche micro du câble de liaison entre le magnétophone et l'ordinateur. Mettez alors le magnétophone en position "Enregistrement" (Record) et faites votre annonce sonore.

Une fois cette annonce faite, arrêtez le magnétophone, rebranchez la fiche micro, et vous pouvez procéder normalement à l'enregistrement de votre programme.

Lorsqu'ensuite vous recherchez ce programme sur la bande, vous n'avez qu'à lire les annonces sonores précédant chaque programme pour le retrouver rapidement. Un autre truc consiste, une fois le programme enregistré, à compléter l'annonce par une information telle que : "Le prochain programme se trouve à (numéro de compteur de la fin du programme)". De la sorte, lors d'une recherche, vous savez jusqu'où doit se faire le défilement rapide suivant.

Les cassettes de "99" contiennent toutes des annonces sonores, afin de vous faciliter le chargement des programmes.

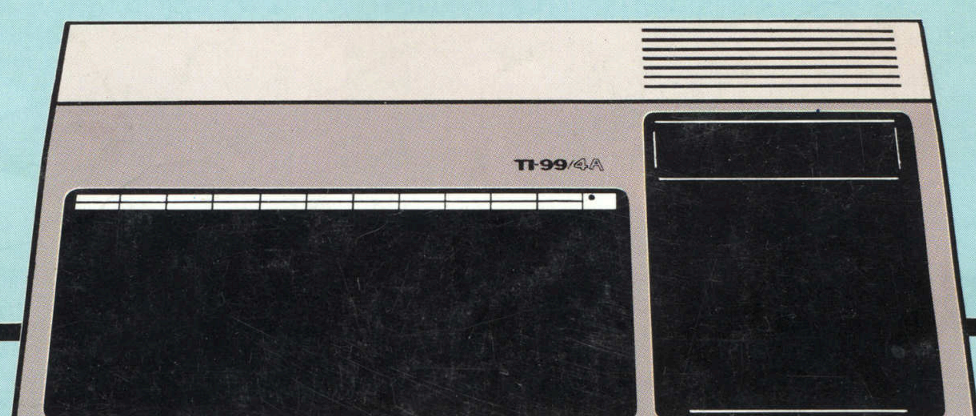
Suite de la page 60

```
1980 REM REDEFINITION DES CODES ASCII DES CA
RACTERES A,H,I,M,N,O,R,U
1990 CALL CHAR(35,"003844447C444444")
2000 CALL CHAR(36,"00446C5454444444")
2010 CALL CHAR(37,"00446464544C4C44")
2020 CALL CHAR(38,"0044444444444438")
2030 CALL CHAR(40,"004444447C444444")
2040 CALL CHAR(41,"0078444478504844")
2050 CALL CHAR(42,"007C44444444447C")
2060 CALL CHAR(43,"0038101010101038")
2070 CALL CHAR(44,"00446C5454444444")
2080 REM COULEUR DES CARACTERES
2090 CALL COLOR(1,9,8)
2100 CALL COLOR(2,13,8)
2110 REM AFFICHAGE DES CARACTERES
2120 CALL HCHAR(2,18,40)
2130 CALL HCHAR(3,19,35)
2140 CALL HCHAR(3,21,36)
2150 CALL HCHAR(3,23,37)
2160 CALL HCHAR(4,20,41)
2170 CALL HCHAR(4,22,42)
2180 CALL HCHAR(4,24,43)
2190 CALL HCHAR(5,25,38)
2200 CALL HCHAR(6,26,44)
2210 GOTO 390
2220 REM INDICATION DE L'OCTAVE: POSITION DE
 L'INDEX (^)
2230 CALL HCHAR(6,2,32,12)
2240 CALL HCHAR(6,2+2*J,112)
2250 RETURN
2260 REM OBTENTION DU SON ET INDICATION DE L
A TOUCHE PRESSEE (PAVES):
2270 REM TOUCHES BLANCHES
2280 CALL SOUND(-300,F1(I,J),2,F2(I,J),6)
2290 CALL HCHAR(20,X1,128)
2300 GOTO 1220
2310 REM TOUCHES NOIRES
2320 CALL SOUND(-340,F1(I,J),2,F2(I,J),6)
2330 CALL HCHAR(15,X2,120)
2340 CALL HCHAR(15,X2+1,121)
2350 GOTO 1220
2360 REM EFFACEMENT DES PAVES SUR LES TOUCHE
S BLANCHES ET NOIRES
2370 CALL HCHAR(20,X1,97)
2380 CALL HCHAR(15,X2,99,2)
2390 GOTO 1240
```

CASSETTE D'ACCOMPAGNEMENT DISPONIBLE

99 MAGAZINE

10 PROGRAMMES DE JEU
LA SAUVEGARDE DES FICHIERS
LES LUTINS DU LOGO
L'ORDINATEUR DE L'AVENIR...



NUMÉRO 2 - Trimestriel - Septembre 1983 - 40 F

99 MAGAZINE

Sommaire

Page

Editorial par Hervé Thiriez	**5**
Micro-monde LOGO : les lutins par Sophie	**7**
La Bataille Navale par Alain Dupont	**12**
Le Crocodile Savant par Claude Lamy	**15**
Jeu du Simon par Georges-Henri Pelard	**17**
Jeu des Carrés par Didier Bicking	**24**
L'ordinateur de l'avenir par Didier Bicking	**27**
TI-Writer à l'essai par Thomas Coppens	**33**
Canon de Purcell par Claire Arribet	**35**
Chicken Helper par Claude Lamy	**39**
La sauvegarde des fichiers et programmes par Jean Sahal	**41**
Le Mastermind par Edouard Jouan	**42**
Othello par Didier Bicking	**47**
Astéroïdes par Edouard Jouan	**52**
Kermit la grenouille par Alexandre Duback	**54**
Le Biorythme par Thomas Coppens	**57**
Micro-informations	**61**
Courrier des lecteurs	**65**
Trucs et Astuces	**52 - 60**

Les annonceurs
Shift Edition : page 2 - La règle à calcul : page 6 - Editions du PSI : page 16 - Guide de l'O.I. : page 46 - Duriez : page 63 - SICOB : page 64 - SECF : page 67 - Texas Instruments : page 68.

Editions MEV - 49 rue Lamartine - 78000 Versailles - Tél : (3) 951.24.43
(Dépôt légal : 3ᵉ trimestre 1983)

Editorial

Voici déjà le numéro deux de notre revue; le nombre de lecteurs de "99" augmente régulièrement : nous vous remercions de la confiance que vous nous faites ainsi. Nous devons vous adresser nos excuses pour les quelques cafouillages que nous connaissons encore; nous espérons que ce seront les derniers.

Plusieurs de nos auteurs sont des enseignants; certains ont été piégés par la douce euphorie des vacances et n'ont pu nous livrer leurs articles à temps. L'échéance du Sicob ne nous a pas permis d'attendre leurs articles. Pour nous faire pardonner, nous avons mis plein de programmes de jeu dans la revue, sur les cassettes 99-2 (cassette d'accompagnement du numéro 2) et 99-C, la troisième cassette séparée que nous proposons.

Dans ce numéro, vous trouverez de très nombreux programmes ludiques ou éducatifs, dont plusieurs proviennent du club TiSoft-Home. **Didier Bicking** *a laissé libre cours à son imagination dans un article de "science fiction" sur l'ordinateur de l'avenir; il nous transmet aussi un programme d'Othello (ou Reversi) et un jeu des carrés.*

Dans le registre musical, **Claire Arribet** *nous donne un splendide canon (très pacifique) de Purcell et* **Georges-Henri Pelard** *présente sa version du Simon à la fois en TI Basic et en Extended Basic. Les joueurs pourront encore se défouler avec les contributions de* **Edouard Jouan** *(le Mastermind et les Astéroïdes), la Bataille Navale* **d'Alain Dupont** *ou la Grenouille* **d'Alexandre Duback**.

Les anxieux pourront se rassurer ou s'angoisser, selon la date, avec le programme de Biorythme amené par **Thomas Coppens** *qui analyse aussi pour nous TI-Writer.* **Sophie** *nous explique l'apport des "sprites" au Logo du TI-99.* **Jean Sahal** *nous aide à mieux utiliser le magnétophone pour la sauvegarde des fichiers et des programmes. Enfin,* **Claude Lamy** *analyse pour nous le Crocodile Savant et Chicken Helper. Une nouvelle rubrique, Micro-informations, vous met au courant des nouveautés.*

Les lecteurs qui ont des articles et/ou des programmes originaux à nous envoyer peuvent le faire, si possible en envoyant les programmes sur cassette ou disquette. Les articles sont rétribués au tarif de 150 francs la page imprimée. Nous sommes à la recherche de contributions originales.

Nous vous rappelons que la cassette d'accompagnement du numéro 1, ainsi que toute autre cassette, peut être acquise séparément. Enfin si, comme nous l'espérons, vous appréciez "99", abonnez-vous et faites connaître notre revue : le service que nous pouvons vous rendre est d'autant plus grand que vous serez nombreux. En tous cas, bonne lecture et amusez-vous bien avec tous les programmes de ce numéro 2 !

Enfin, nous sommes heureux de vous faire savoir que les petites annonces provenant de particuliers seront gratuites dans « 99 magazine ». Nous n'en avions pas dans le numéro 1, par la force des choses... Nous attendons par conséquent vos annonces éventuelles. En ce qui concerne le Courrier des Lecteurs, nous nous efforcerons de répondre au fur et à mesure, dans la limite de nos moyens. N'oubliez pas à cet égard de joindre une enveloppe timbrée pour la réponse !

Hervé Thiriez

Rédacteur en chef - Directeur de la publication : Hervé Thiriez. **Comité de rédaction :** Didier Bicking, Thomas Coppens, Roger Didi, Gérard Vantraille. **Dessins :** Laurent Bidot. **Ont collaboré à ce numéro :** Claire Arribet, Didier Bicking, Thomas Coppens, Alexandre Duback, Alain Dupont, Edouard Jouan, Claude Lamy, Georges-Henri Pelard, Jean Sahal, Sophie.

Editions MEV - 49 rue Lamartine - 78000 Versailles. Tél. (3) 951.24.43.

Régie Publicitaire : Force 7, Anne Jourdan, 39 rue de la Grande-aux-Belles, 75483 Paris Cedex 10. Tél. : (1) 238.66.10.

Diffusion auprès des boutiques et libraires : Editions du PSI, 44-51, rue Jacquard, BP 86, 77400 Lagny-sur-Marne.

Composition : Télécompo, 13-15 avenue du Petit Parc, 94300 Vincennes.

Impression : Imprimerie Rosay, 47, avenue de Paris, 94300 Vincennes. Tél. : (1) 328.18.63.

Micro-Monde Logo : les lutins

Sophie

Vous avez pu lire, dans le numéro 1 de "99 magazine", un article très documenté brossant un panorama rapide des possibilités et de l'intérêt de la version LOGO disponible sur le TI 99/4A. Nous vous proposons, cette fois-ci, une promenade dans un monde peu connu, celui des lutins (sprites).

Seul le TI-99 offre actuellement cette possibilité : les possesseurs d'Apple peuvent se bricoler une carte à lutins, en maniant le fer à souder et à partir d'un schéma donné dans le Spécial LOGO de Byte en Août 82 ... à moins que, paresseux et fortunés, ils ne préfèrent faire un voyage express aux Etats-Unis pour acheter le Sprite-Board de Krell.

Les utilisateurs du Basic Etendu ou de l'assembleur du TI 99/4A ont peut-être déjà eu la curiosité d'utiliser les lutins. Les domaines d'application de ce micro-monde concernent le mouvement et la couleur, par exemple la modélisation en dynamique (voilà qui devrait intéresser les professeurs de physique), ou l'animation et, pourquoi pas, les jeux vidéo. C'est une occasion privilégiée de s'exercer à la simulation et aux effets de trucage.

Avoir un lutin et lui parler

Gérard Bossuet, un des pionniers du LOGO en France, définit un lutin comme "un petit être très pudique qui ne consent à se montrer que décemment vêtu". D'une manière moins poétique, mais probablement plus parlante pour les adultes, nous décrirons un lutin comme un objet disponible sur l'écran de visualisation, auquel on peut fournir un certain nombre d'attributs tels que forme, couleur, emplacement et, éventuellement, cap et vitesse.

L'appel d'un lutin

Il convient tout d'abord d'attirer l'attention d'un lutin, en quelque sorte d'établir une espèce de communication téléphonique avec lui. On dispose de 32 lutins, qui sont désignés par leur numéro. Par exemple :

APPELLE LUTIN 1

met l'utilisateur en relation avec le premier lutin. *Remarque : chacune des instructions, comme celle qui précède, doit être validée en appuyant sur la touche ENTER pour être exécutée par la machine. Nous ne le mentionnerons plus.*

Si, au cours du dialogue avec l'ordinateur, vous avez oublié le numéro du lutin avec lequel vous conversez, vous pouvez retrouver facilement cette information en tapant :

AFFICHE QUI

la réponse vous sera alors donnée :

LE LUTIN 1

Cette indication est parfois fort utile.

Le choix d'une forme

On attribue au lutin une forme, qui sera en général dénotée par un nombre :

FIXEFORME 3

L'instruction précédente donne la forme 3, une forme de fusée, au lutin numéro 1.

Attention ! Le numéro d'appel du lutin n'a aucun lien a priori avec son numéro de forme, de la même façon que votre numéro de téléphone n'a rien à voir avec votre numéro de Sécurité Sociale ou l'immatriculation de votre voiture.

Cinq formes "préfabriquées" sont disponibles d'origine sur votre ordinateur; elles portent les numéros 1 à 5 ; ce sont les mêmes formes qui définissent les lutins initiaux en Basic Etendu. Les 27 autres formes possibles sont définissables par vous, comme nous le montrerons plus loin dans cet article.

Le choix d'une couleur

Après avoir donné une forme à notre lutin (éventuellement l'une des cinq), il nous reste à lui attribuer une couleur. Seize couleurs différentes peuvent être utilisées, que l'on peut indifféremment désigner par un code numérique ou par leur nom.

0	Transparent	8	Roux
1	Noir	9	Orange
2	Vert	10	Jaune
3	Tilleul	11	Citron
4	Azur	12	Olive
5	Ciel	13	Violet
6	Rouge	14	Gris
7	Bleu	15	Blanc

NDLR : c'est tout à fait par hasard que le Citron suit le Tilleul dans ce tableau.

Pour colorer en rouge la fusée avec laquelle nous sommes actuellement "en communication", il suffit de taper soit :

FIXECOULEUR :ROUGE

soit :

FIXECOULEUR 3

Nous verrons dans un article ultérieur à quoi correspond le caractère ":" précédant chaque nom de couleur.

Le déplacement du lutin

En tapant "CENTRE", nous amenons maintenant au centre de l'écran notre fusée habillée en rouge.

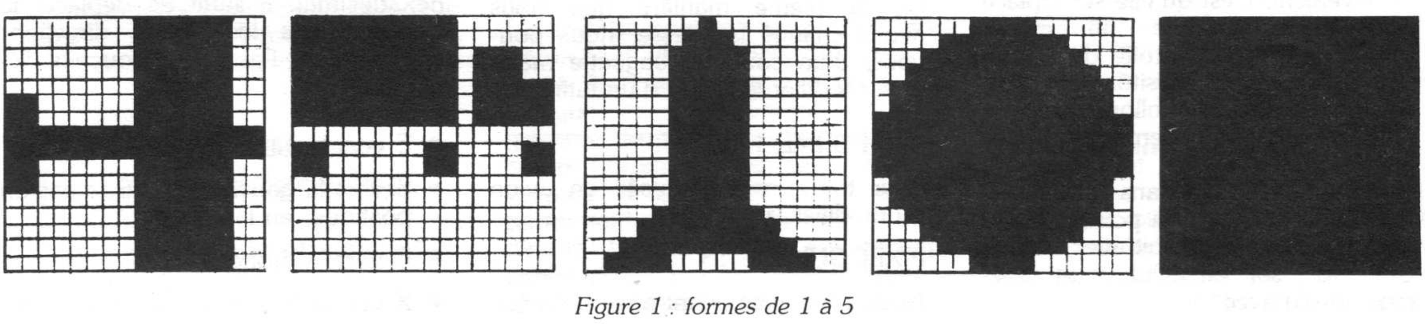


Figure 1 : formes de 1 à 5

En conclusion, nous avons suivi la séquence suivante :

APPELLE LUTIN
Met le 1er lutin en activité

FIXEFORME 3
Lui donne la forme 3 (fusée)

FIXECOULEUR : ROUGE
Peint la fusée en rouge

CENTRE
Amène le lutin au centre

Nous pouvons maintenant tester diverses instructions de déplacement, illustrées de gauche à droite sur la figure 2 :

AVANCE 50

fait un bond vers le haut de l'écran

DROITE 90

n'a pas d'effet immédiat visible ; mais si nous donnons maintenant l'ordre

AVANCE 70

nous constatons que le bond s'effectue maintenant vers la droite de l'écran.

FIXECAP 0

Le "cap" est l'angle que fait le regard invisible du lutin avec le Nord, mesuré en degrés et compté positivement dans le sens des aiguilles d'une montre. Ainsi, quand le lutin "pointe" vers le haut, les ordres "GAUCHE 90", "DROITE -90" (pour les gens compliqués), "DROITE 270", "FIXECAP 270" et "FIXECAP -90" sont-ils équivalents : tout à l'Ouest !

Avec le cap 0, la fusée s'élève à présent d'une manière plus conforme à la réalité, disparait en haut de l'écran et revient par le bas.

Modification des attributs

Les attributs donnés au lutin peuvent à tout moment être modifiés. On peut ralentir celui-ci avec :

FIXEVITESSE 1

ou l'accélérer avec :

FIXEVITESSE 100

Quelle est la vitesse maximale d'un lutin ? Vous pouvez la trouver par tâ-

mauve, qui se comporte plutôt come un hélicoptère en train d'atterrir !

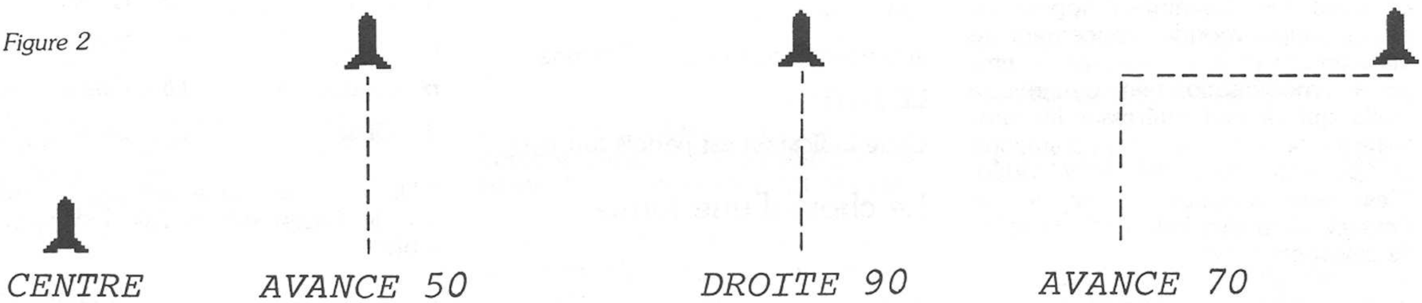
Si nous ne nous rappelons plus précisément des ordres que nous avons donnés au lutin, nous pouvons facilement retrouver ces informations :

AFFICHE CAP
0
AFFICHE VITESSE
-5
AFFICHE FORME
1
AFFICHE COULEUR
13

Création de formes de lutins

Délaissons à présent les engins aéronautiques, et cherchons d'autres domaines d'inspiration. Pourquoi ne pas donner à un lutin la forme d'un chat ? Pour cela, il nous faut d'abord déterminer le numéro de code que nous allons attribuer à cette nouvelle forme (supérieur à 5). Prenons par exemple le numéro 10 et tapons :

Figure 2



Ce comportement surprendra les utilisateurs habitués à la tortue, qui s'attendent à voir le lutin changer d'orientation avec "DROITE 90".

Toutefois, ces déplacements par petits bonds ne sont pas très satisfaisants pour rendre le mouvement d'une fusée, qu'on souhaiterait plutôt voir se déplacer de façon plus régulière. Ceci est possible, en attribuant au lutin une vitesse de déplacement :

FIXEVITESSE 10

Et voilà la fusée en marche ! Le seul inconvénient, c'est qu'elle se déplace actuellement comme un crabe, s'échappant vers la droite de l'écran pour réapparaître aussitôt à la gauche de l'écran. N'oublions pas que nous avons précédemment, avec l'ordre "DROITE 90", tourné le "regard" de notre lutin dans cette direction. Nous pouvons lui pointer le nez en l'air de nouveau, soit avec "GAUCHE 90", soit en utilisant un repérage absolu avec :

tonnements, et grâce à un peu de réflexion. Quand cette vitesse maximale est dépassée, vous obtenez le message suivant :

FIXEVITESSE N'AIME PAS RECEVOIR ...

suivi du nombre que vous avez essayé.

La marche arrière est possible avec par exemple :

FIXEVITESSE -5

Comment arrêtez-vous le lutin ? A vous de jouer ...

De la même manière que nous avons changé la vitesse, nous pouvons, d'un coup de baguette magique, modifier la forme d'un lutin :

FIXEFORME 1

pour transformer la fusée en avion ou modifier la couleur :

FIXECOULEUR 13

Nous voici maintenant avec un avion

CREEFORME 10

Là, l'écran change radicalement d'aspect : le fond, qui avait une couleur bleu ciel, est devenu vert ; de plus, une grille de 16 carreaux sur 16 occupe la presque totalité de la page. Le petit avion mauve, quant à lui, continue à se promener imperturbablement de haut en bas.

Dans le coin supérieur gauche de l'écran clignote un petit carré noir : ce curseur mobile va nous permettre de dessiner la forme du lutin. Pas besoin ici de se triturer les méninges (comme en TI-Basic) à calculer en hexadécimal. Il suffit de déplacer le curseur dans les quatre directions cardinales à l'aide des touches fléchées :

- E vers le haut (Elever)
- S vers la gauche (Sinistre : pas en politique, en latin)
- D vers la droite (Droite)
- X vers le bas (mort aux X ?)

Remarquez que, comme ces touches portent sur leur face avant des flèches de direction, les moyens mnémotechniques indiqués ci-dessus ne sont pas indispensables.

En se déplaçant, le curseur ne laisse pas de trace ... sauf si vous appuyez sur la touche FCTN : dans ce cas, en quittant une case, le curseur y laisse un carré noir. Vous allez voir que c'est véritablement un jeu d'enfant que de dessiner ce petit chat.

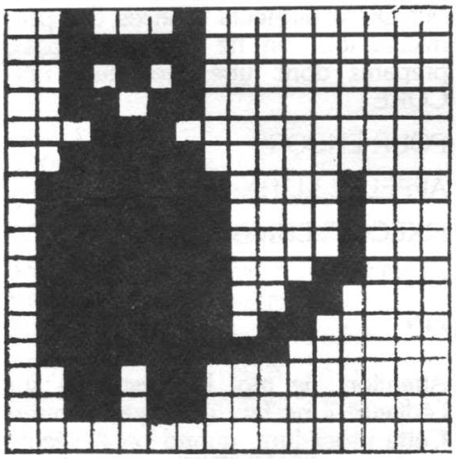


Figure 3 - forme 10

Quand vous avez fini, appuyez sur "FCTN 9" pour enregistrer cette forme dans la mémoire de l'ordinateur. Elle y restera disponible tant que vous n'aurez pas éteint l'appareil ni quitté LOGO. Si vous désirez conserver cette forme pour une utilisation ultérieure, il suffit de l'enregistrer sur une cassette ou sur une disquette.

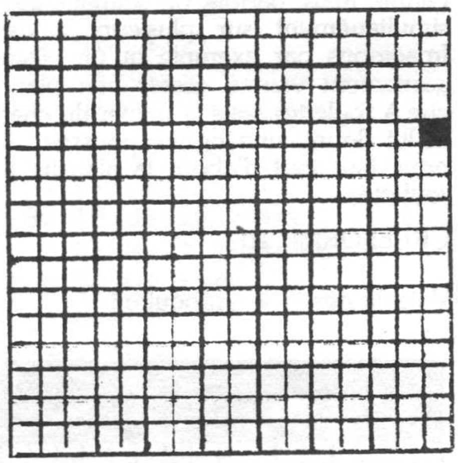


Figure 4 : avant

Si vous avez mal reproduit le dessin prévu, rien de plus simple que d'en modifier la forme. Il suffit de repasser, **sans** maintenir la touche FCTN enfoncée, sur les cases que vous désirez "blanchir". Il vous arrivera peut-être aussi de "perdre" le curseur, au début de l'utilisation de l'éditeur de formes, tant que vous ne serez pas habitué à la structure torique de la grille. En effet, si l'on quitte une ligne par la droite, le curseur revient par la gauche (voir figure 4) après avoir, à toute vitesse, "fait le tour par l'arrière de l'écran".

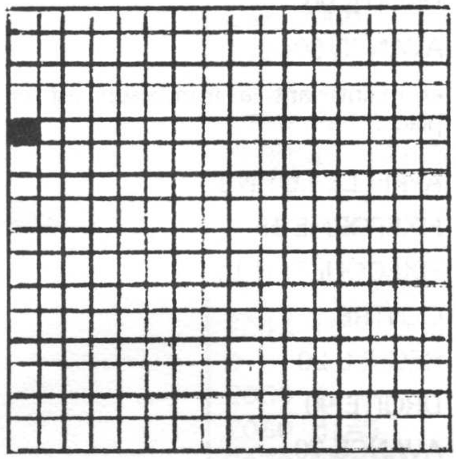


Figure 4 : après

Ce comportement inattendu du curseur provoque toujours un effet de surprise au début, chez les adultes comme chez les enfants. Mais on tire très vite parti de cette particularité pour simplifier certains déplacements, en évitant ainsi de repasser sur une partie du dessin déjà achevée.

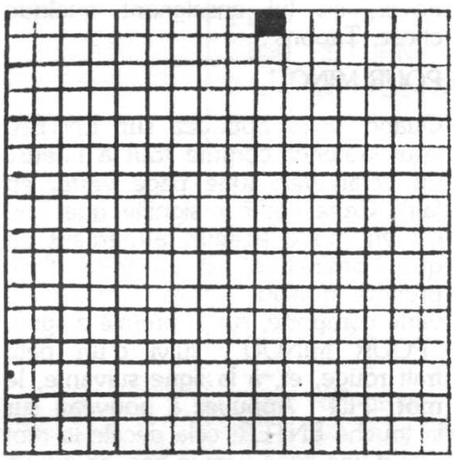


Figure 5 : avant

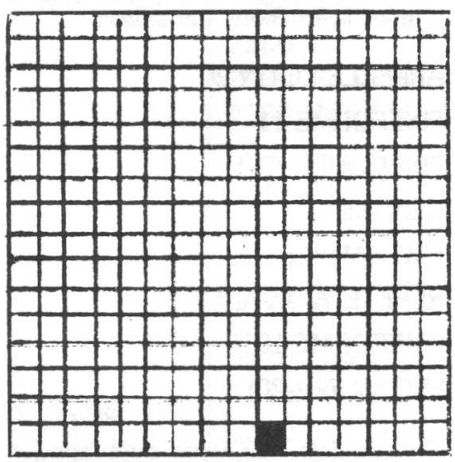


Figure 5 : après

Maintenant que notre dextérité augmente, nous pouvons réaliser, sous le numéro 15, la forme suivante :

Nous sommes enfin prêts à passer à l'étape suivante.

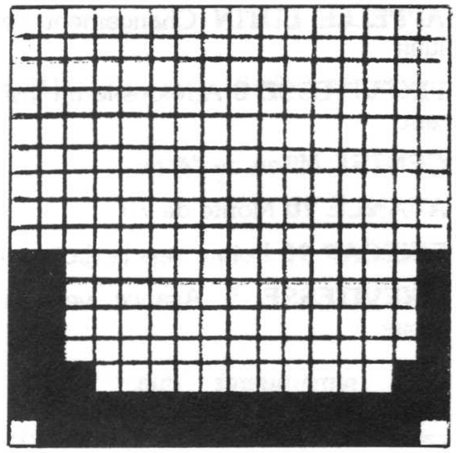


Figure 6 : forme 15

Création d'un dessin animé

Nous allons reproduire sur l'écran un paysage rudimentaire comportant quelques arbres et un soleil. Ensuite, nous y ajouterons un avion (qui vole, bien sûr !), puis nous mettrons le chat dans un coin et nous lui donnerons des instructions pourqu'il aille s'installer dans son panier.

Mais, me direz-vous, le chat que nous venons de créer est un chat assis ! D'accord, alors créons vite un chat debout sous la forme 11.

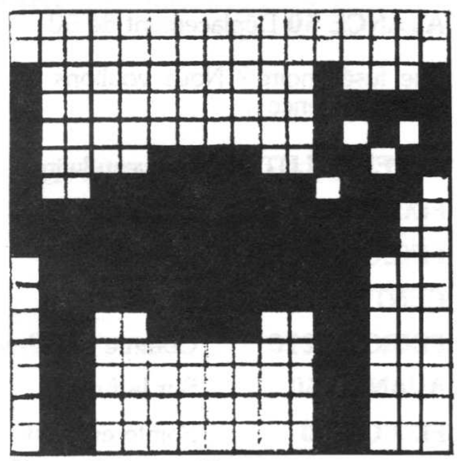


Figure 7 : forme 11

Prêt ? Allons-y ! Commençons par le soleil.

APPELLE LUTIN 3 Troisième lutin

FIXEFORME 4 Forme de balle

CENTRE Milieu de l'écran

AVANCE 80 Monte de 80

GAUCHE 90 Quart de tour gauche

AVANCE 110 Va en haut à gauche

FIXECOULEUR 10 Le soleil est jaune

Nous avons maintenant un beau soleil doré dans le coin supérieur gauche de l'écran. Occupons-nous à présent de l'avion, qui est pour le moment toujours en mouvement sur l'écran.

APPELLE LUTIN Changement de lutin

FIXEVITESSE 0 Arrêtons-le un instant

CENTRE Milieu de l'écran

AVANCE 70 Monte de 70

FIXECAP 90 Pointe vers la droite

FIXEVITESSE 5 Avance vers la droite

Vous remarquerez que, lorsque l'avion passe aux abords du soleil, il cache une partie du lutin soleil, ce qui est conforme à la réalité : par rapport au spectateur, l'avion est devant le soleil.

Plaçons les arbres dans notre paysage.

APPELLE LUTIN 7 Changement de lutin

FIXEFORME 3 La forme d'une fusée

FIXECOULEUR 1 La fusée est noire

CENTRE

FIXECAP -90 Direction à gauche

AVANCE 50 Déplacement de 50

Une fusée noire ? Nous voulions un arbre ! Patience ...

APPELLE LUTIN 6 Nouveau lutin

FIXEFORME 4 Une balle

FIXECOULEUR 2 ... verte

CENTRE

FIXECAP 270 Comme "-90"

AVANCE 50 Sur la fusée

FIXECAP 0 Pointe en haut

AVANCE 14 Monte de 14

Et voilà un arbre ! En voulez-vous un autre ?

APPELLE LUTIN 10

FIXEFORME 3

FIXECOULEUR 8

CENTRE

AVANCE 10

DROITE 90

AVANCE 30

Voici un tronc roux ...

APPELLE LUTIN 9

FIXEFORME 4

FIXECOULEUR 12

CENTRE

AVANCE 24

DROITE 90

AVANCE 30

et maintenant la frondaison est en place.

APPELLE LUTIN 15

FIXEFORME 15

FIXECOULEUR 14

CENTRE

RECULE 20

DROITE 90

AVANCE 70

Le panier, gris, est installé dans le coin inférieur droit de l'écran. Nous avons aussi appris à utiliser un nouvel ordre, "RECULE".

Jusqu'à présent, nous avions seulement dialogué avec l'ordinateur; en fait, nous l'avons "piloté" en lui donnant des ordres. Ici, nous allons innover, en lui enseignant quelque chose. Tapons :

POUR MINOU

Quand vous appuyez sur ENTER, vous obtenez comme tout à l'heure un écran vert (une page verte, en fait). Cette couleur signifie que l'on est en mode éditeur, autrement dit que l'ordinateur est en train d'apprendre quelque chose. Cette page verte comporte, sur la première ligne, "POUR MINOU", suivi d'un petit trait rouge, et, à la ligne suivante, le mot "FIN". Appuyez à nouveau sur la touche ENTER; cela décale le mot FIN d'une ligne vers le bas, ce qui va nous permettre d'incorporer une à une les instructions suivantes :

APPELLE LUTIN 20

FIXEFORME 11

FIXECOULEUR 1

CENTRE

FIXECAP 105

REPETE 72 AVANCE 1

FIXEFORME 10

FIXEXY 72 (-20)

Chaque fois que vous avez appuyé sur ENTER, le mot FIN est descendu d'une ligne. Maintenant, appuyons sur "FCTN 9". Nous retrouvons la page bleue, mais de chat, pas la moindre trace !

Comme il y a beaucoup de texte sur l'écran, et que tout cela n'a pas l'air bien net, tapons "EFFACEECRAN" (tel quel, en un mot avec deux "E" au milieu). Tout le texte disparait, mais les lutins ne sont pas affectés par cet ordre.

Tapons maintenant "MINOU". Ca y est, le chat arrive sur l'écran, se dirige tranquillement vers son panier, et s'y installe confortablement. Pour un peu, on s'attendrait à l'entendre ronronner ...

Pour vous permettre d'utiliser MINOU à plusieurs reprises (les premières fois, on ne s'en lasse pas), préparez donc une procédure ENCORE :

POUR ENCORE

APPELLE LUTIN 20

FIXECOULEUR 0

CENTRE

FIN

Attention : le mot FIN est déjà sur l'éditeur; il ne faut donc pas l'entrer. Cette procédure, quand on l'exécute en tapant "ENCORE" ... fait disparaître le chat, un véritable tour de magie ! Il suffit alors de taper "MINOU" à nouveau pour retrouver notre chat se dirigeant vers son panier.

Plusieurs lutins à la fois

Jusqu'ici, nous nous sommes toujours adressés aux lutins l'un après l'autre. Mais, parfois, on souhaite agir simultanément sur plusieurs lutins. Imaginons par exemple qu'un vilain garnement vienne glisser une planche à roulettes sous le panier du chat et lui donne une bonne poussée ! Il nous faut tout d'abord la planche à roulettes :

CREEFORME 10

Voici le dessin à reproduire :

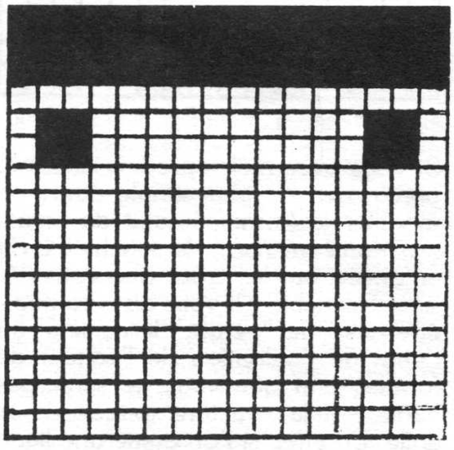


Figure 8 : forme 12

Une fois le dessin terminé, revenons en mode dialogue avec "FCTN 9", puis lançons-nous :

POUR BOUGE

APPELLE LUTIN 5

FIXEFORME 12

FIXECOULEUR 6

CENTRE

FIXEXY 70 (-36)

APPELLE [5 10 20]

FIXECAP -90

FIXEVITESSE 3

FIN

Ici, nous avons demandé aux trois lutins de faire la même chose; mais on peut également donner à chacun des instructions différentes en fonction de son code (son "numéro de téléphone").

Bien sûr, il s'agit là d'un dessin animé rudimentaire : c'est à peu près le genre de réalisation à laquelle aboutissent des élèves provenant des toutes premières années de l'école primaire, en une dizaine d'heures, sans l'aide d'adultes.

Passe-temps ou activité éducative ?

J'entends déjà d'ici des commentaires peu enthousiastes tels que "Tout ça, c'est bien joli, mais ce n'est pas sérieux. Nous n'envoyons pas nos enfants à l'école pour qu'ils y perdent leur temps à de telles balivernes !". Peut-être faut-il souligner tout le travail pédagogique qui entoure une réalisation de ce type.

Construction de formes

La construction de formes fournit des activités qui se rattachent à une partie importante (tant par le volume que par la qualité des connaissances à maîtriser) du programme de Mathématiques du Cours Préparatoire :

- utilisation de la comptine des nombres pour effectuer un dénombrement de carreaux à noircir, lorsque les enfants recopient à l'écran une forme qu'ils ont préalablement dessinée sur papier ;

- distinction entre le comptage des carreaux et le comptage des bords de carreaux (vous vous souvenez des fameux problèmes des piquets et des intervalles entre piquets) ;

- spatialisation : au-dessus, au-dessous, à droite, à gauche, en haut, en bas, près, loin, ...

- addition et soustraction : si j'ai six carreaux noirs sur une rangée, il **doit** m'en rester dix blancs (total de seize), sinon je ferais bien de recompter ...

Création de scénario

La création du scénario que l'on va réaliser à l'écran fait travailler des notions extrêmement diverses liées au récit, à la chronologie, aux relations dans l'espace (perspective par exemple, au moins au sens de la succession ordonnée de plusieurs plans, correspondant à des profondeurs plus ou moins grandes dans le champ).

Prise en compte de l'espace

Le positionnement des différents éléments du conte amène aussi à d'autres types de prise en compte de l'espace : utiliser la totalité de la surface de l'écran pour y disposer agréablement les divers lutins ne se réussit pas du premier coup. Il y aura beaucoup de tâtonnements, des essais, des modifications, l'apprentissage d'une technique de résolution par approximations successives. Les valeurs à donner pour la vitesse des lutins en mouvement sont l'occasion d'une sensibilisation à la comparaison de nombres (pour les plus âgés, ce sera aussi l'observation de vitesses apparentes et réelles d'objets physiques).

Alors, effectivement, l'instituteur ou l'institutrice dont les élèves préparent un tel dessin animé ne leur "enseigne" rien, au sens où il ne transmet pas de connaissances prédéterminées, sous la forme d'un cours directif. Et pourtant, les élèves apprennent beaucoup de choses ... tout en éprouvant du plaisir, ce qui surprendra peut-être ceux pour qui l'enseignement doit forcément avoir un aspect rébarbatif.

Il ne s'agit pourtant pas d'une école de la facilité, et la mise au point d'un projet par les élèves leur demande patience, effort et ténacité. Mais à quoi sert l'enseignant dans un tel cadre, et ne pourrait-on à la limite s'en passer, et imaginer des écoles où trois cent élèves passeraient leurs journées sur autant d'ordinateurs individuels ?

A notre sens, la présence de l'enseignant demeure irremplaçable : son rôle n'est pas tant celui d'un transmetteur de connaissances (depuis des années déjà, les connaissances sont accessibles en dehors de l'Ecole, grâce aux bibliothèques, à la télévision, ...) que celui d'une personne-ressource, capable de favoriser le plein épanouissement des capacités de chacun, c'est-à-dire d'amener chaque élève le plus loin possible dans son propre parcours sur le chemin du savoir.

NDLR : la plupart des ordres du LOGO possèdent une définition abrégée. Ainsi, taper "AV" a le même effet que de taper "AVANCE". Afin de préserver la clarté du texte, nous n'avons pas fait apparaître les abréviations dans cet article d'initiation, celles-ci pouvant être trouvées sans difficulté dans la documentation du LOGO.

Sophie est le pseudonyme d'un groupe de membres de l'Association G.R.E.P.A.C.I.F.I.C. qui vous proposeront, dans les numéros à venir, de parcourir quelques aspects de l'approche LOGO, en offrant des présentations générales, comme dans ce numéro, ainsi que des témoignages vécus.

G.R.E.P.A.C.I.F.I.C. (Groupe de Recherche et d'Etudes pour une Pratique Active et Coordonnée de l'Informatique en Formation Initiale ou Continue) est une association sans but lucratif (loi de 1901) dont le but principal est l'établissement de liens de communication entre les personnes (chercheurs, enseignants, industriels, parents, ...) intéressées par l'approche LOGO.

L'association assure la diffusion d'informations concernant l'approche LOGO. Les cahiers PACIFIC, de périodicité et volume variables, constituent des témoignages d'activités menées sur le terrain (compte-rendus d'expérimentation, mémoires de recherche, ...). Les Documents de Travail sont destinés à alimenter la réflexion sur des thèmes plus ponctuels, souvent encore au stade du défrichage.

L'association organise des ateliers d'initiation à l'approche LOGO, principalement durant les vacances scolaires, mais le cas échéant en réponse à une demande précise, sur un thème spécifique ou pour un organisme particulier.

L'association entretient des relations avec les équipes de recherche LOGO de nombreux pays étrangers.

La bataille navale

Alain Dupont

Le programme que je vous présente ici est une version adaptée pour TI-99 du jeu bien connu de la Bataille Navale. Ici, l'ordinateur n'est pas un adversaire : il dispose les navires, réclame les tirs et en présente les effets.

On commence, bien entendu, par effacer l'écran (ligne 100), puis on lui donne une couleur (ligne 110) et on détermine la couleur des caractères qui seront imprimés (lignes 120 à 140). On marque ensuite le titre. Deux méthodes sont possibles. La première consiste à le marquer au bas de l'écran par un ordre PRINT et à le faire monter au milieu de l'écran par une suite d'ordres PRINT vides de texte.

J'ai choisi la deuxième méthode qui me semble plus élégante. Elle consiste à composer directement le titre au milieu de l'écran au moyen d'une série d'ordres CALL HCHAR. Ainsi, la ligne 110 dispose la lettre "B" sur la ligne numéro 11, colonne 10 de l'écran, la ligne 120 dispose la lettre "A" sur la ligne 11, colonne 11, etc. La ligne 230 dispose les deux "L" du mot "BATAILLE". Le mot "NAVALE", quant à lui, est écrit à partir de la ligne 15, colone 18 (lignes 250 à 300).

Les lignes 310 à 330 ne posent pas de problème particulier : l'enfoncement de n'importe quelle touche donne la valeur 1 à la variable ETAT, ou la valeur -1 si la touche reste enfoncée. Dans chacun de ces deux cas, le programme est envoyé à la ligne 370.

L'exposé des règles (lignes 370 à 420) est fait par une impression selon la première méthode : cela aurait été trop long de procéder comme pour le titre. Mais si vous y tenez, vous pouvez toujours modifier cette partie du programme. Comme toujours en informatique, il n'existe pas de solution unique à un problème donné...



Intervient ensuite une phase d'initialisation. Je ne vais pas ici vous la détailler entièrement, mais seulement commenter certaines des procédures que j'ai utilisées. Remarquez en premier lieu les lignes 510 et 520. L'instruction OPTION BASE 1 concerne les tableaux de variables. Si, comme dans la ligne 520, on dimensionne un tableau MAT(9,9), les lignes et colonnes de ce tableau sont numérotées de 1 à 9, c'est-à-dire que ce tableau pourra comporter 81 valeurs différentes. Si j'avais écrit OPTION BASE 0 (ou si je n'avais pas mis cette ligne 510), alors la première ligne et la première colonne du tableau auraient été numérotées 0, ce qui est quand même moins clair qu'une numérotation commençant par 1. Il en va de même pour tous les autres tableaux du programme, tant que l'instruction OPTION BASE n'est pas réutilisée.

L'ordinateur place ensuite un navire vertical de trois cases, puis un navire horizontal de même valeur, en s'assurant qu'il reste toujours une distance d'au moins une case entre les deux navires. De la même manière, il place les quatre vedettes d'une case chacune, distantes d'au moins une case entre elles ou avec les navires. Enfin, il construit le diagramme dans lequel le joueur va envoyer ses tirs.

Pour demander le numéro de ligne et celui de la colonne dans lesquelles le joueur veut tirer, j'ai utilisé là encore la deuxième méthode d'impression, ce qui permet de maintenir les images fixes à l'écran. Sans cela, le diagramme aurait fini par disparaître en haut de celui-ci, ce qui aurait présenté certains inconvénients...

Enfin, l'analyse des tirs et l'affichage des résultats ne présente pas de difficulté (lignes 2250 à 2710). La ligne 2340 a pour objet de renvoyer à l'initialisation du tir si le coup indiqué a déjà été fait : on ne peut tirer deux fois dans la même case et, dans ce cas, le compteur de tirs n'est pas incrémenté.

Au niveau de la stratégie pure, l'intérêt de ce programme est assez limité : j'ai conçu ce programme pour mon fils de 6 ans et 1/2 dans le but de le familiariser avec le repérage dans le plan par le système ligne-colonne. En revanche, pour ce qui est de la technique de programmation, "Bataille Navale" me semble assez complet. Il comporte la manipulation de tableaux matriciels et utilise abondamment les IF THEN ELSE. Tous les sous-programmes disponibles sur le TI-99, à l'exception de CALL JOYST, y sont utilisés et vous pouvez aisément en analyser les fonctions.

```
10 REM ***************
20 REM * BATAILLE    *
30 REM *   NAVALE    *
40 REM ***************
50 REM
60 REM ==================
70 REM COPYRIGHT 99 MAGAZINE
80 REM ET ALAIN DUPONT
90 REM ==================
100 CALL CLEAR
110 CALL SCREEN(11)
120 FOR V=1 TO 8
130 CALL COLOR(V,14,11)
140 NEXT V
```

```
150 REM   ***************
160 REM   *  TITRE      *
170 REM   ***************
180 CALL HCHAR(11,10,66)
190 CALL HCHAR(11,11,65)
200 CALL HCHAR(11,12,84)
210 CALL HCHAR(11,13,65)
220 CALL HCHAR(11,14,73)
230 CALL HCHAR(11,15,76,2)
240 CALL HCHAR(11,17,69)
250 CALL HCHAR(15,18,78)
260 CALL HCHAR(15,19,65)
270 CALL HCHAR(15,20,86)
280 CALL HCHAR(15,21,65)
290 CALL HCHAR(15,22,76)
300 CALL HCHAR(15,23,69)
310 PRINT "APPUYER SUR N'IMPORTE QUELLE
TOUCHE POUR COMMENCER"
320 CALL KEY(0,TOUCHE,ETAT)
330 IF ETAT=0 THEN 320 ELSE 370
340 REM   ****************
350 REM   * REGLE DU JEU *
360 REM   ****************
370 CALL CLEAR
380 PRINT "IL FAUT COULER DEUX NAVIRES D
E 3 CASES ET 4 VEDETTES     D'UNE CASE"
390 PRINT ::::
400 PRINT "APPUYER SUR N'IMPORTE QUELLE
TOUCHE"
410 CALL KEY(0,TOUC,ET)
420 IF ET=0 THEN 410 ELSE 430
430 CALL CLEAR
440 CALL SCREEN(12)
450 FOR V2=1 TO 8
460 CALL COLOR(V2,2,12)
470 NEXT V2
480 REM   *******************
490 REM   * INITIALISATIONS *
500 REM   *******************
510 OPTION BASE 1
520 DIM MAT(9,9),NAV(8)
530 Z=TIR+COUP
540 TIR=0
550 COUP=0
560 REM   ****************
570 REM   * CARACTERES   *
580 REM   ****************
590 CALL CHAR(128,"FF818181818181FF")
600 REM *CAR. COUP AU BUT*
610 CALL CHAR(130,"FFFFFFFFFFFFFFFF")
620 REM *CAR.COUP A L'EAU*
630 CALL CHAR(129,"FF81A59999A581FF")
640 REM *CAR. LETTRE L *
650 CALL CHAR(152,"00808080808080F8")
660 PRINT "JE PLACE LES NAVIRES SUR LE D
IAGRAMME"
670 REM   *****************
680 REM   *     TI 99     *
690 REM   *     PLACE     *
700 REM   *   LE NAVIRE   *
710 REM   *   VERTICAL    *
720 REM   *****************
730 RANDOMIZE
740 N1=INT(RND*9)+1
750 N2=INT(RND*9)+1
760 IF N1<8 THEN 770 ELSE 810
770 MAT(N1,N2)=1
780 MAT(N1+1,N2)=1
790 MAT(N1+2,N2)=1
800 GOTO 970
810 IF N1=8 THEN 820 ELSE 860
820 MAT(N1,N2)=1
830 MAT(N1-1,N2)=1
840 MAT(N1+1,N2)=1
850 GOTO 970
860 MAT(N1,N2)=1
870 MAT(N1-1,N2)=1
880 MAT(N1-2,N2)=1
890 REM *****************
900 REM *   TI 99       *
910 REM * PLACE   LE    *
920 REM * NAVIRE HORIZ. *
930 REM *****************
940 REM
950 REM  * C'EST PLUS DUR!*
960 REM
970 RANDOMIZE
980 N3=INT(9*RND)+1
990 N4=INT(9*RND)+1
1000 IF N3=N1 THEN 1010 ELSE 1020
1010 IF N4=N2 THEN 970
1020 IF N4<8 THEN 1030 ELSE 1220
1030 IF MAT(N3,N4)=1 THEN 970
1040 IF MAT(N3,N4+1)=1 THEN 970
1050 IF MAT(N3,N4+2)=1 THEN 970
1060 IF N4<7 THEN 1070 ELSE 1080
1070 IF MAT(N3,N4+3)=1 THEN 970
1080 IF N4>1 THEN 1090 ELSE 1100
1090 IF MAT(N3,N4-1)=1 THEN 970
1100 IF N3>1 THEN 1110 ELSE 1140
1110 IF MAT(N3-1,N4)=1 THEN 970
1120 IF MAT(N3-1,N4+1)=1 THEN 970
1130 IF MAT(N3-1,N4+2)=1 THEN 970
1140 IF N3<9 THEN 1150 ELSE 1180
1150 IF MAT(N3+1,N4)=1 THEN 970
1160 IF MAT(N3+1,N4+1)=1 THEN 970
1170 IF MAT(N3+1,N4+2)=1 THEN 970
1180 MAT(N3,N4)=1
1190 MAT(N3,N4+1)=1
1200 MAT(N3,N4+2)=1
1210 GOTO 1540
1220 IF N4=8 THEN 1230 ELSE 1390
1230 IF MAT(N3,N4)=1 THEN 970
1240 IF MAT(N3,N4+1)=1 THEN 970
1250 IF MAT(N3,N4-1)=1 THEN 970
1260 IF MAT(N3,N4-2)=1 THEN 970
1270 IF N3>1 THEN 1280 ELSE 1310
1280 IF MAT(N3-1,N4-1)=1 THEN 970
```

```
1290 IF MAT(N3-1,N4)=1 THEN 970
1300 IF MAT(N3-1,N4+1)=1 THEN 970
1310 IF N3<9 THEN 1320 ELSE 1350
1320 IF MAT(N3+1,N4)=1 THEN 970
1330 IF MAT(N3+1,N4-1)=1 THEN 970
1340 IF MAT(N3+1,N4+1)=1 THEN 970
1350 MAT(N3,N4)=1
1360 MAT(N3,N4-1)=1
1370 MAT(N3,N4+1)=1
1380 GOTO 1540
1390 IF MAT(N3,N4)=1 THEN 970
1400 IF MAT(N3,N4-1)=1 THEN 970
1410 IF MAT(N3,N4-2)=1 THEN 970
1420 IF N3>1 THEN 1430 ELSE 1460
1430 IF MAT(N3-1,N4)=1 THEN 970
1440 IF MAT(N3-1,N4-1)=1 THEN 970
1450 IF MAT(N3-1,N4-2)=1 THEN 970
1460 IF N3<9 THEN 1470 ELSE 1500
1470 IF MAT(N3+1,N4)=1 THEN 970
1480 IF MAT(N3+1,N4-1)=1 THEN 970
1490 IF MAT(N3+1,N4-2)=1 THEN 970
1500 IF MAT(N3,N4-3)=1 THEN 970
1510 MAT(N3,N4)=1
1520 MAT(N3,N4-1)=1
1530 MAT(N3,N4-2)=1
1540 GOTO 1600
1550 REM  *****************
1560 REM  *     TI 99     *
1570 REM  *     PLACE     *
1580 REM  * 4 VEDETTES    *
1590 REM  *****************
1600 FOR G=1 TO 7 STEP 2
1610 RANDOMIZE
1620 NAV(G)=INT(RND*9)+1
1630 NAV(G+1)=INT(RND*9)+1
1640 IF MAT(NAV(G),NAV(G+1))=1 THEN 1610
1650 IF NAV(G+1)<9 THEN 1660 ELSE 1670
1660 IF MAT(NAV(G),NAV(G+1)+1)=1 THEN 16
10
1670 IF NAV(G+1)>1 THEN 1680 ELSE 1690
1680 IF MAT(NAV(G),NAV(G+1)-1)=1 THEN 16
10
1690 IF NAV(G)<9 THEN 1700 ELSE 1720
1700 IF MAT(NAV(G)+1,NAV(G+1))=1 THEN 16
10
1710 IF NAV(G)>1 THEN 1720 ELSE 1730
1720 IF MAT(NAV(G)-1,NAV(G+1))=1 THEN 16
10
1730 MAT(NAV(G),NAV(G+1))=1
1740 NEXT G
1750 REM ****************
1760 REM * CONSTRUCTION *
1770 REM *     DU       *
1780 REM * DIAGRAMME    *
1790 REM ****************
1800 CALL CLEAR
1810 CALL COLOR(13,7,16)
1820 CALL HCHAR(5,8,128,9)
1830 FOR I=0 TO 8
1840 CALL VCHAR(5,8+I,128,9)
1850 NEXT I
1860 FOR L=0 TO 8
1870 CALL HCHAR(5+L,7,65+L)
1880 NEXT L
1890 FOR R=1 TO 9
1900 CALL HCHAR(4,7+R,48+R)
1910 NEXT R
1920 GOSUB 2530
1930 REM  ****************
1940 REM  * APPEL LIGNE  *
1950 REM  ****************
1960 CALL HCHAR(15,4,152)
1970 CALL HCHAR(15,5,73)
1980 CALL HCHAR(15,6,71)
1990 CALL HCHAR(15,7,78)
2000 CALL HCHAR(15,8,69)
2010 CALL HCHAR(15,9,63)
2020 CALL KEY(0,KEY,STAT)
2030 IF STAT=0 THEN 2020
2040 IF KEY>73 THEN 2020
2050 IF KEY<65 THEN 2020
2060 FOR S=1 TO 9
2070 IF KEY=64+S THEN 2090
2080 NEXT S
2090 I1=S
2100 CALL HCHAR(15,11,64+S)
2110 REM  *****************
2120 REM  * APPEL COLONNE *
2130 REM  *****************
2140 CALL HCHAR(17,4,67)
2150 CALL HCHAR(17,5,79)
2160 CALL HCHAR(17,6,76)
2170 CALL HCHAR(17,7,79)
2180 CALL HCHAR(17,8,78,2)
2190 CALL HCHAR(17,10,69)
2200 CALL HCHAR(17,11,63)
2210 CALL KEY(0,KE,STA)
2220 IF STA=0 THEN 2210
2230 IF KE>57 THEN 2210
2240 IF KE<49 THEN 2210
2250 FOR T=1 TO 9
2260 IF KE=48+T THEN 2280
2270 NEXT T
2280 I2=T
2290 CALL HCHAR(17,13,48+T)
2300 REM
2310 REM  *DANS L'EAU*
2320 REM
2330 GOSUB 2620
2340 IF MAT(I1,I2)=0 THEN 2350 ELSE 2430
2350 CALL HCHAR(4+I1,7+I2,129)
2360 CALL SOUND(200,2000,0)
2370 Z=Z+1
2380 GOSUB 2690
2390 GOTO 1920
2400 REM
2410 REM  *AU BUT*
2420 REM
```

```
2430 CALL HCHAR(4+I1,7+I2,130)
2440 CALL SOUND(500,-5,0)
2450 TIR=TIR+1
2460 Z=Z+1
2470 IF TIR<10 THEN 2490
2480 END
2490 GOTO 2380
2500 REM **********
2510 REM *COMPTEUR*
2520 REM **********
2530 DIZ=INT(Z/10)
2540 UNITE=Z-DIZ*10
2550 CALL HCHAR(8,22,84)
2560 CALL HCHAR(8,23,73)
2570 CALL HCHAR(8,24,82)
2580 CALL HCHAR(8,25,83)
2590 CALL HCHAR(10,24,48+DIZ)
2600 CALL HCHAR(10,25,48+UNITE)
2610 RETURN
2620 CALL GCHAR(4+I1,7+I2,W)
2630 IF W=129 THEN 2640 ELSE 2670
2640 CALL SOUND(-500,-3,0)
2650 GOSUB 2690
2660 GOTO 1960
2670 IF W=130 THEN 2640
2680 RETURN
2690 CALL HCHAR(15,4,32,8)
2700 CALL HCHAR(17,4,32,12)
2710 RETURN
```

Le crocodile savant

Claude Lamy

Le Crocodile Savant est un module de programme commercialisé par Texas Instruments. Il s'agit d'un programme de jeu éducatif.

Principe du jeu

Un crocodile est représenté dans le coin en bas à droite de l'écran. Une pomme apparaît sur la gauche et se dirige vers l'animal alors qu'un nombre s'inscrit dans son abdomen. Une opération arithmétique simple (addition ou soustraction de deux nombres à un seul chiffre) figure dans la pomme. Si le nombre figurant sur le crocodile est le résultat de l'opération, il faut appuyer sur la barre d'espacement. La gueule du saurien s'ouvre alors et il mange la pomme. Si, en revanche, le nombre ne correspond pas au résultat, il ne faut rien faire.

Au bout de dix essais, un deuxième crocodile se dessine au dessus du précédent, un peu plus vers la gauche. Les pommes s'approchent donc plus rapidement de lui et le joueur doit décider plus vite d'ouvrir ou non la gueule de l'animal. Il en va ainsi jusqu'à la représentation du cinquième crocodile.

Si, à un certain niveau, le joueur ne donne pas un nombre suffisant de bonnes réponses, le jeu redescend au niveau précédent, c'est-à-dire que le dernier crocodile s'efface et on revient à la ligne en-dessous.

Le jeu dure un certain temps, déterminé à l'avance par le joueur lui-même lors de la mise en route du programme. Au bout de ce délai, le résultat est affiché ainsi que le score maximal obtenu par l'un quelconque des joueurs depuis le lancement du programme.

Module ou programme ?

A voir fonctionner, c'est très beau, mais cela vaut-il le prix d'achat d'un module ? Je m'explique : un programme écrit en Basic peut tout aussi bien remplir les mêmes fonctions, même si son graphisme est moins spectaculaire. Encore que représenter un crocodile ne soit pas du tout hors de portée d'un programmeur moyennement entraîné ! Certains "gadgets" poseront sans doute quelques problèmes, comme par exemple la détermination au départ de la vitesse de déplacement des pommes (déterminant un niveau de difficulté.). Mais cela est loin d'être insurmontable. Un bon programmeur disposant du Basic Etendu peut en faire autant, sinon mieux.

Conclusion

En conclusion, je vous conseillerais plutot d'écrire vous-même un programme pour réaliser cette fonction de test mathématique. Même si les dessins en sont moins jolis, vous trouverez sans doute d'autres "trucs" pour l'agrémenter de manière à le rendre attractif pour vos enfants. Mais ce module du Crocodile savant est un premier pas dans une nouvelle politique de programmes : les programmes de jeux éducatifs. Nous ne saurions trop conseiller à Texas Instruments de persister dans cette voie tout à fait conforme à la vocation familiale du TI-99, mais en nous proposant quand même des modules un peu plus élaborés.



Jeu du Simon

Georges-Henri Pelard

Le "Simon" est un grand classique des jeux sur micro-ordinateur. Rappelons pour ceux qui l'ignorent que le but du jeu consiste à reproduire une séquence sonore de plus en plus longue composée de quatre notes de hauteurs distinctes. Pour vous aider à mener à bien cette "dictée musicale", chaque note jouée est associée au clignotement d'une surface colorée, chaque couleur correspondant bien évidemment à une hauteur et une seule. La version présentée ici est capable de mémoriser jusqu'à cent notes. On y joue à l'aide des manettes de jeu, ou des touches E, X, S et D du clavier; à deux, ou seul contre l'ordinateur qui se charge de générer de façon aléatoire la séquence.

Initialisation

Les lignes 25 et 30 réservent en mémoire la place nécessaire à l'utilisation de :

1 - Deux cents variables de type P(M,N) pouvant prendre n'importe quelle valeur entière comprise entre 1 et 4 inclus. Chacune de ces variables correspond à l'une des quatre notes générées par vous ou votre adversaire, ce que l'ordinateur distingue grâce à l'indice M en lui attribuant la valeur 1 ou 2. Ainsi, à P(M,N) est associée la Nième note de la séquence reproduite par le joueur M.

2 - Cinq constantes chaînes A$(J) utilisées pour faciliter l'écriture du programme lors de la redéfinition des 20 caractères qui composent le dessin du "Simon" (lignes 50 à 95).

3 - Cinq variables B(I) représentant les cinq meilleurs scores.

4 - Cinq variables B correspondant chacune aux trois initiales de ceux qui les ont réalisés. Les lignes 35 à 45 constituent pour ces variables une "remise à zéro" plus élégante que la chaîne vide qui leur serait sans elle attribuée.

Menu

Les lignes 120 à 165 affichent le menu. Toutes les couleurs utilisées sont définies dans quatre sous-programmes (lignes 1160 à 1300). Cela n'est pas indispensable, mais on peut ainsi simuler, en jouant des "transparences", le résultat d'un PRINT AT malheureusement absent en TI Basic.

En faisant apel au sous-programme CALL KEY (lignes 1305 à 1350), les lignes 170 à 230 vont déterminer la suite des opérations à effectuer. Une pression sur "AID" provoquera l'affichage des règles du jeu, alors qu'en appuyant sur les touches 1 ou 2, ou sur le bouton de tir des manettes de jeu, on amènera le programme au début de la partie. Toute autre action sur le clavier entraîne l'affichage des cinq meilleurs scores (sous-programme contenu dans les lignes 1585 à 1645), puis le retour au menu.

Règles du jeu

Les lignes 235 à 415 affichent en quatre "écrans" une description sommaire des règles du jeu. Ceux-ci sont séparés par l'appel du sous-programme CALL KEY qui remplace ici une boucle vide, avantageusement puisqu'une simple pression sur une touche quelconque du clavier annule le "délai" ainsi créé.

Affichage du jeu

Les lignes 420 à 650 ne présentent rien de particulier. On y utilise simplement quatre boucles et la possibilité offerte en TI Basic de répéter horizontalement ou verticalement un même caractère, afin d'afficher le plus rapidement possible les quelques 260 petits carrés qui composent ce graphisme.

Début de partie

A l'aide d'une nouvelle utilisation du sous-programme CALL KEY, les lignes 665 à 710 vous permettent soit de visualiser les cinq meilleurs scores lorsque vous enfoncez la touche "ERASE", soit de commencer à jouer (pression sur "BEGIN" ou le bouton de tir.). Toute autre action vous renvoie au menu. Cette partie du programme n'est certes pas indispensable, mais elle évite que votre TI-99 ne commence sans vous lorsque vous jouez contre lui.

Jeu

A la ligne 730 commence véritablement le jeu. Cinq variables sont alors définies : M et N dont nous avons déjà parlé, D qui représente la durée de chaque note de la séquence, DE qui détermine le temps de réponse auquel vous avez droit pour créer ou répéter chacune d'entre elles, et SC, votre score, affiché en permanence en haut de l'écran par les lignes 755 à 770. Le déroulement du jeu dépend ensuite de la version choisie. Si vous jouez seul (KM = 49), l'ordinateur attribue de façon aléatoire à la variable P(M,N) une valeur qui va définir, dans le sous-programme des lignes 1390 à 1420, un branchement vers l'un des quatre sous-programmes responsables des notes et clignotements (lignes 1425 à 1480).

Si vous jouez à deux (KM = 50), c'est le joueur 1 qui va déterminer lui-même, à l'aide du clavier et/ou des manettes, vers quel sous-programme s'opérera le branchement. Le programme utilise pour cela les lignes 815 à 940 et boucle sur lui-même tant que votre réponse est correcte (test de la ligne 930) et que vous n'avez pas atteint la centième note de la séquence (test de la ligne 960).

Fin de partie

Le programme ne s'aventure ici que si vous avez perdu. Il vous rejoue une dernière fois la séquence qui vous y a amené, en profite pour remettre toutes les variables P(M,N) à zéro (lignes 1010 à 1035), et vous invite, quoiqu'il arrive, à rejouer.

Meilleurs scores

Amener le programme aux lignes 1080 à 1155 est ce qu'il peut vous arriver de mieux. Cela se produit lorsque vous venez de réaliser l'un des cinq meilleurs scores. L'ordinateur les classe par ordre décroissant, vous demande d'entrer vos initiales (lignes 1140 et 1145), affiche les scores, puis retourne à la ligne 1050 pour vous inviter à rejouer.

Sous-programmes

Les quatre sous-programmes des lignes 1160 à 1300 définissent les couleurs utilisées.

Le sous-programme CALL KEY (lignes 1305 à 1350) scrute le clavier

et vérifie que le bouton de tir de la manette (celle que vous utilisez pour jouer seul) n'a pas été enfoncé. Si c'est le cas, ou si une pression est exercée sur une touche quelconque du clavier, ou encore si le compteur A atteint sa valeur limite DE, alors on quitte le sous-programme.

Celui des lignes 1355 à 1385 efface les commentaire qui apparaissent au centre du dessin du "Simon".

Les lignes 1390 à 1420 répètent la dernière séquence sonore jouée par votre adversaire. En cours de jeu, on fait appel à ce sous-programme en appuyant sur "AID" ou sur le bouton de tir.

Chacun des quatre sous-programmes suivants (lignes 1425 à 1580) génère une note et efface tous les caractères appartenant à un même groupe en leur attribuant la couleur 1 (transparent).

les lignes 1585 à 1645 affichent les cinq meilleurs scores pendant une durée déterminée par la variable DE.

Le dernier sous-programme (lignes 1650 à 1750) affiche les commentaires, caractère par caractère, lorsque l'on ne peut utiliser un PRINT qui ferait "monter" tous les caractères déjà présents à l'écran.

Les instructions DATA contiennent les lignes, colonnes et codes des caractères à afficher.

Basic Etendu

Cette version ne diffère que très peu de la précédente. Le résultat à l'écran est le même, mais la rapidité d'exécution est accrue.

On y utilise plusieurs instructions par ligne dont le très pratique PRINT AT qui permet de supprimer quelques sous-programmes d'affichage, ainsi qu'ACCEPT AT, d'utilisation beaucoup plus souple qu'INPUT. Enfin, puisque le Basic traite les variables et les constantes de la même façon et que l'heure était à la compilation, les constantes A$(I) ont été supprimées, remplacées remplacées par les variables B$(I) déjà utilisées.

Programme Simon - TI Basic

```
1 REM ***************************
2 REM *    SIMON EN TI-BASIC     *
3 REM * COPYRIGHT 99 MAGAZINE    *
4 REM * Georges-Henri PELARD *
5 REM ***************************
10 REM ====================
15 REM Initialisation et définition des
caractères
20 REM  ====================
25 OPTION BASE 1
30 DIM P(2,100),A$(5),B(5),B$(5)
35 FOR I=1 TO 5
40 B$(I)="---"
45 NEXT I
50 A$(1)="FFFFFFFFFFFFFFFF"
55 A$(2)="0103070F1F3F7FFF"
60 A$(3)="80C0E0F0F8FCFEFF"
65 A$(4)="FFFEFCF8F0E0C080"
70 A$(5)="FF7F3F1F0F070301"
75 FOR I=95 TO 119 STEP 8
80 FOR J=1 TO 5
85 CALL CHAR(I+J,A$(J))
90 NEXT J
95 NEXT I
100 CALL CHAR(91,"3C4299A1A199423C")
105 REM ====================
110 REM Affichage de l'écran d'introduct
ion (menu)
115 REM ====================
120 GOSUB 1280
125 CALL SCREEN(2)
130 CALL CLEAR
135 GOSUB 1175
140 PRINT TAB(4);"APPUYER SUR": :TAB(9);
"1 POUR 1 JOUEUR": :TAB(9);"2 POUR 2 JOU
EURS"
145 PRINT : : : :TAB(4);CHR$(91)&" 99 MA
GAZINE & GH PELARD"
150 LI=44
155 RESTORE 1690
160 GOSUB 1665
165 GOSUB 1245
170 DE=300
175 GOSUB 1320
180 KM=K
185 IF (K=49)+(K=50)+(KT=18)THEN 435
190 D=20
195 FOR I=1 TO 5
200 FOR J=1 TO 4
205 ON J GOSUB 1440,1480,1520,1560
210 NEXT J
215 NEXT I
220 IF (K=1)+(K=55)THEN 250
225 GOSUB 1600
230 GOTO 130
235 REM ===================
240 REM    Règles du jeu
245 REM ===================
250 CALL CLEAR
255 GOSUB 1175
260 PRINT " DANS SA VERSION ""1 JOUEUR""
": :"LE JEU CONSISTE A REPRODUIRE"
265 PRINT :"  UNE SEQUENCE DE NOTES DE":
:"PLUS EN PLUS LONGUE GENEREE"
270 PRINT :" DE FACON ALEATOIRE PAR L'"
275 PRINT :TAB(10);"ORDINATEUR."
280 GOSUB 1245
285 GOSUB 1320
290 CALL CLEAR
295 GOSUB 1175
300 PRINT " DANS SA SECONDE VERSION LE":
 :"  PRINCIPE EST LE MEME MAIS": :"   LA
 SEQUENCE EST ALORS"
305 PRINT :"   GENEREE PAR LES DEUX": :"
ADVERSAIRES QUI L'AUGMENTENT": :" D'UNE
NOTE A CHAQUE FOIS."
```

```
310 GOSUB 1245
315 GOSUB 1320
320 CALL CLEAR
325 GOSUB 1175
330 PRINT "CHAQUE NOTE EST ASSOCIEE A":
:" UNE SURFACE COLOREE QUI"
335 PRINT :"  CLIGNOTE LORSQU'ELLE EST"
340 PRINT :TAB(12);"JOUEE.": :"CHAQUE SU
RFACE EST ASSOCIEE"
345 PRINT :"A UNE DIRECTION (HAUT, BAS,"
: :" DROITE OU GAUCHE) QUE LE"
350 PRINT :" JOUEUR DETERMINE SOIT EN":
:"APPUYANT SUR LES TOUCHES E,": :"X,D OU
 S, SOIT EN UTILISANT"
355 PRINT :TAB(5);"LES MANETTES DE JEU."
360 GOSUB 1245
365 DE=500
370 GOSUB 1320
375 CALL CLEAR
380 GOSUB 1175
385 PRINT "EN COURS DE JEU, POUR FAIRE":
 :"REPETER LA DERNIERE SEQUENCE"
390 PRINT :"IL SUFFIT D'APPUYER SUR AID"
: :"OU D'ACTIONNER LE BOUTON DE"
395 PRINT :TAB(7);"TIR DES MANETTES."
400 GOSUB 1245
405 DE=250
410 GOSUB 1320
415 GOTO 130
420 REM ====================
425 REM Affichage de l'écran du jeu
430 REM ====================
435 CALL CLEAR
440 GOSUB 1210
445 FOR I=0 TO 4
450 CALL HCHAR(1+I,15-I,96,2*I+5)
455 CALL VCHAR(10-I,28-I,104,2*I+5)
460 CALL HCHAR(23-I,15-I,112,2*I+5)
465 CALL VCHAR(10-I,6+I,120,2*I+5)
470 NEXT I
475 FOR I=0 TO 3
480 CALL HCHAR(1+I,14-I,97)
485 CALL HCHAR(1+I,20+I,98)
490 CALL HCHAR(6+I,25+I,106)
495 CALL HCHAR(15+I,28-I,107)
500 CALL HCHAR(20+I,23-I,115)
505 CALL HCHAR(20+I,11+I,116)
510 CALL HCHAR(15+I,6+I,124)
515 CALL HCHAR(6+I,9-I,121)
520 NEXT I
525 FOR I=0 TO 2
530 CALL HCHAR(5+I,11+I,100)
535 CALL HCHAR(5+I,23-I,99)
540 CALL HCHAR(6+I,24-I,105)
545 CALL HCHAR(16+I,22+I,108)
550 CALL HCHAR(17+I,21+I,114)
555 CALL HCHAR(17+I,13-I,113)
560 CALL HCHAR(6+I,10+I,122)
565 CALL HCHAR(18-I,10+I,123)
570 NEXT I
575 FOR I=0 TO 1
580 CALL HCHAR(6,7*I+13,96,2)
585 CALL HCHAR(7-I,14+I,99)
590 CALL HCHAR(6+I,19+I,100)
595 CALL VCHAR(7*I+8,23,104,2)
600 CALL HCHAR(9+I,22+I,108)
605 CALL HCHAR(14+I,23-I,105)
610 CALL HCHAR(18,7*I+13,112,2)
615 CALL HCHAR(17+I,14+I,114)
620 CALL HCHAR(17+I,20-I,113)
625 CALL VCHAR(7*I+8,11,120,2)
630 CALL HCHAR(9+I,12-I,123)
635 CALL HCHAR(14+I,11+I,122)
640 NEXT I
645 GOSUB 1280
650 REM ====================
655 REM   début de partie
660 REM ====================
665 LI=15
670 RESTORE 1725
675 GOSUB 1665
680 DE=300
685 GOSUB 1320
690 IF 1+(K=7)+(K=51)THEN 705
695 GOSUB 1600
700 GOTO 435
705 IF 1+(KT=18)+(K=14)+(K=53)THEN 130
710 GOSUB 1370
715 REM ====================
720 REM         Jeu
725 REM ====================
730 M=1
735 N=1
740 D=500-INT(104*LOG(N))
745 DE=100+INT(200/N)
750 SC=Z+10*(N-1)
755 CALL HCHAR(1,30,48)
760 CALL HCHAR(1,29,48+N-100*INT(N/100)-
10*INT(N/10))
765 CALL HCHAR(1,28,48+INT(N/10)-10*INT(
N/100))
770 CALL HCHAR(1,27,48+INT(N/100))
775 IF KM=50 THEN 810
780 M=1
785 RANDOMIZE
790 P(2,N)=INT(RND*4)+1
795 GOSUB 1405
800 CALL HCHAR(12,17,43)
805 GOTO 815
810 CALL HCHAR(12,17,48+M)
815 FOR O=1 TO N
820 A=0
825 CALL KEY(M,KT,S)
830 CALL KEY(3,K,S)
835 CALL JOYST(M,X,Y)
840 IF (KT=18)+(K=1)+(K=55)THEN 880
845 IF (K=69)+(Y=4)THEN 890
850 IF (K=68)+(X=4)THEN 900
```

```
855 IF (K=88)+(Y=-4)THEN 910
860 IF (K=83)+(X=-4)THEN 920
865 A=A+1
870 IF A=DE THEN 990
875 GOTO 825
880 GOSUB 1405
885 GOTO 815
890 P(M,O)=1
895 GOTO 925
900 P(M,O)=2
905 GOTO 925
910 P(M,O)=3
915 GOTO 925
920 P(M,O)=4
925 IF P(2/M,O)=0 THEN 935
930 IF P(M,O)<>P(2/M,O)THEN 990
935 ON P(M,O)GOSUB 1440,1480,1505,1560
940 NEXT O
945 GOSUB 1370
950 M=2/M
955 N=N+1
960 IF N<100 THEN 740
965 Z=Z+1000
970 GOTO 1010
975 REM ===================
980 REM  Fin de Partie
985 REM ===================
990 GOSUB 1370
995 LI=8
1000 RESTORE 1735
1005 GOSUB 1665
1010 D=20
1015 FOR I=1 TO N
1020 ON 1+P(2/M,I)GOSUB 1460,1440,1480,1
520,1560
1025 P(1,I)=0
1030 P(2,I)=0
1035 NEXT I
1040 IF N=100 THEN 710
1045 IF SC>B(5)THEN 1095
1050 LI=20
1055 RESTORE 1740
1060 GOSUB 1665
1065 DE=200
1070 GOSUB 1320
1075 IF (K=15)+(K=57)THEN 130 ELSE 435
1080 REM ===================
1085 REM  Meilleurs Scores
1090 REM ===================
1095 CALL CLEAR
1100 FOR I=1 TO 4
1105 IF SC>B(I)THEN 1115
1110 NEXT I
1115 FOR J=5 TO I+1 STEP -1
1120 B$(J)=B$(J-1)
1125 B(J)=B(J-1)
1130 NEXT J
1135 B(I)=SC
1140 INPUT "VOS INITIALES ? ":I$
1145 B$(I)=SEG$(I$,1,3)
1150 GOSUB 1600
1155 GOTO 1050
1160 REM ===================
1165 REM  Transparence 1
1170 REM ===================
1175 FOR I=1 TO 8
1180 CALL COLOR(I,1,1)
1185 NEXT I
1190 RETURN
1195 REM ===================
1200 REM  Transparence 2
1205 REM ===================
1210 FOR I=9 TO 12
1215 CALL COLOR(I,1,1)
1220 NEXT I
1225 RETURN
1230 REM ===================
1235 REM   Cyan / Noir
1240 REM ===================
1245 FOR I=1 TO 8
1250 CALL COLOR(I,8,1)
1255 NEXT I
1260 RETURN
1265 REM ===================
1270 REM   Colorisation
1275 REM ===================
1280 CALL COLOR(9,7,1)
1285 CALL COLOR(10,11,1)
1290 CALL COLOR(11,5,1)
1295 CALL COLOR(12,13,1)
1300 RETURN
1305 REM ===================
1310 REM      Call Key
1315 REM ===================
1320 A=0
1325 CALL KEY(3,K,S)
1330 CALL KEY(1,KT,ST)
1335 A=A+1
1340 IF A=DE THEN 1350
1345 IF (S=0)*(KT<>18)THEN 1325
1350 RETURN
1355 REM ===================
1360 REM  Effacage Central
1365 REM ===================
1370 CALL HCHAR(10,14,32,9)
1375 CALL HCHAR(12,13,32,7)
1380 CALL HCHAR(14,15,32,5)
1385 RETURN
1390 REM ===================
1395 REM <Aid> Répétition
1400 REM ===================
1405 FOR I=1 TO N
1410 ON 1+P(2/M,I)GOSUB 1420,1440,1480,1
520,1560
1415 NEXT I
1420 RETURN
1425 REM ===================
1430 REM  Rouge <349Hz>
```

```
1435 REM ===================
1440 CALL COLOR(9,1,1)
1445 CALL SOUND(D,349,0)
1450 CALL SOUND(D,44730,30)
1455 CALL COLOR(9,7,1)
1460 RETURN
1465 REM ===================
1470 REM  Jaune <440Hz>
1475 REM ===================
1480 CALL COLOR(10,1,1)
1485 CALL SOUND(D,440,0)
1490 CALL SOUND(D,44730,30)
1495 CALL COLOR(10,11,1)
1500 RETURN
1505 REM ===================
1510 REM   Bleu  <587Hz>
1515 REM ===================
1520 CALL COLOR(11,1,1)
1525 CALL SOUND(D,587,0)
1530 CALL SOUND(D,44730,30)
1535 CALL COLOR(11,5,1)
1540 RETURN
1545 REM ===================
1550 REM   Vert  <659Hz>
1555 REM ===================
1560 CALL COLOR(12,1,1)
1565 CALL SOUND(D,659,0)
1570 CALL SOUND(D,44730,30)
1575 CALL COLOR(12,13,1)
1580 RETURN
1585 REM ===================
1590 REM Affichage des scores
1595 REM ===================
1600 CALL CLEAR
1605 GOSUB 1175
1610 PRINT " SCORES :"
1615 FOR I=1 TO 5
1620 PRINT : : :TAB(12);B$(I);"  ";B(I)
1625 NEXT I
1630 GOSUB 1245
1635 DE=300
1640 GOSUB 1320
1645 RETURN
1650 REM ===================
1655 REM  Print At
1660 REM ===================
1665 FOR I=1 TO LI
1670 READ L,C,A
1675 CALL HCHAR(L,C,A)
1680 NEXT I
1685 RETURN
1690 DATA 2,11,98,2,10,96,2,9,96,2,8,97,
3,8,96,4,8,100,4,9,96,4,10,96
1695 DATA 4,11,98,5,11,96,6,11,99,6,10,9
6,6,9,96,6,8,100
1700 DATA 4,13,104,5,13,104,6,13,104
1705 DATA 6,15,112,5,15,112,4,15,112
1710 DATA 4,16,112,4,17,112,5,17,112,6,1
7,112,4,18,112,4,19,114,5,19,112,6,19,11
2
1715 DATA 4,22,120,4,23,122,5,23,120,6,2
3,123,6,22,120,6,21,124,5,21,120,4,21,12
1
1720 DATA 6,25,96,5,25,96,4,25,96,4,26,9
6,4,27,98,5,27,96,6,27,96,2,13,105
1725 DATA 10,14,65,10,15,80,10,16,80,10,
17,85,10,18,89,10,19,69,10,20,90,12,16,8
3
1730 DATA 12,17,85,12,18,82,14,15,66,14,
16,69,14,17,71,14,18,73,14,19,78
1735 DATA 12,13,71,12,14,65,12,15,77,12,
16,69,12,18,48,12,19,86,12,20,69,12,21,8
2
1740 DATA 24,4,65,24,5,80,24,6,80,24,7,8
5,24,8,89,24,9,69,24,10,90,24,12,83,24,1
3,85
1745 DATA 24,14,82,24,16,82,24,17,69,24,
18,68,24,19,48,24,21,48,24,22,85,24,24,6
6
1750 DATA 24,25,65,24,26,67,24,27,75
```

Programme Simon - Extended Basic

```
1 REM *****************************
2 REM *   SIMON EN X-BASIX        *
3 REM *  COPYRIGHT 99 MAGAZINE    *
4 REM * Georges-Henri PELARD *
5 REM *****************************
10 REM =====================
15 REM Initialisation et définition des
caractères
20 REM =====================
25 OPTION BASE 1 :: DIM P(2,100),B(5),B$
(5)
30 B$(1)="FFFFFFFFFFFFFFFF"
35 B$(2)="0103070F1F3F7FFF"
40 B$(3)="80C0E0F0F8FCFEFF"
45 B$(4)="FFFEFCF8F0E0C080"
50 B$(5)="FF7F3F1F0F070301"
55 FOR I=95 TO 119 STEP 8 :: FOR J=1 TO
5 :: CALL CHAR(I+J,B$(J)):: NEXT J :: NE
XT I
60 CALL CHAR(91,"3C4299A1A199423C")
65 FOR I=0 TO 8 :: CALL COLOR(I,8,1):: N
EXT I
70 FOR I=1 TO 5 :: B$(I)="___" :: NEXT I
75 REM =====================
80 REM Ecran d'introduction
(menu)
85 REM =====================
90 CALL SCREEN(2):: GOSUB 730
95 CALL CLEAR
100 RESTORE 1040
105 FOR I=1 TO 44 :: READ L,C,A :: CALL
HCHAR(L,C,A):: NEXT I
110 DISPLAY AT(15,4):"APPUYER SUR" :: DI
SPLAY AT(17,9):"1 POUR 1 JOUEUR" :: DISP
LAY AT(19,9):"2 POUR 2 JOUEURS"
```

```
115 DISPLAY AT(24,4)BEEP:CHR$(91)&" 99 M
AGAZINE & GH PELARD"
120 DE=300
125 GOSUB 755
130 KM=K
135 IF K=49 OR K=50 OR KT=18 THEN 285
140 D=30
145 FOR I=1 TO 5 :: FOR J=1 TO 4 :: ON J
 GOSUB 845,885,925,965 :: NEXT J :: NEXT
 I
150 IF K=1 OR K=55 THEN 180
155 GOSUB 1005
160 GOTO 95
165 REM ====================
170 REM  Règles du Jeu
175 REM ====================
180 DISPLAY AT(10,1)ERASE ALL:"DANS SA V
ERSION ""1 JOUEUR""" :: DISPLAY AT(12,1)
:"LE JEU CONSISTE A REPRODUIRE"
185 DISPLAY AT(14,3):"UNE SEQUENCE DE NO
TES DE" :: DISPLAY AT(16,1):"PLUS EN PLU
S LONGUE GENEREE"
190 DISPLAY AT(18,2):"DE FACON ALEATOIRE
 PAR L'" :: DISPLAY AT(20,10)BEEP:"ORDIN
ATEUR"
195 GOSUB 755
200 DISPLAY AT(10,2)ERASE ALL:"DANS SA S
ECONDE VERSION LE" :: DISPLAY AT(12,3):"
PRINCIPE EST LE MEME MAIS" :: DISPLAY AT
(14,4):"LA SEQUENCE EST ALORS"
205 DISPLAY AT(16,4):"GENEREE PAR LES DE
UX" :: DISPLAY AT(18,1):"ADVERSAIRES QUI
 L'AUGMENTENT" :: DISPLAY AT(20,2)BEEP:"
D'UNE NOTE A CHAQUE FOIS."
210 GOSUB 755
215 DISPLAY AT(2,1)ERASE ALL:"CHAQUE NOT
E EST ASSOCIEE A" :: DISPLAY AT(4,3):"UN
E SURFACE COLOREE QUI"
220 DISPLAY AT(6,3):"CLIGNOTE LORSQU'ELL
E EST" :: DISPLAY AT(8,12):"JOUEE."
225 DISPLAY AT(10,1):"CHAQUE SURFACE EST
 ASSOCIEE" :: DISPLAY AT(12,1):"A UNE DI
RECTION (HAUT, BAS,"
230 DISPLAY AT(14,2):"DROITE OU GAUCHE)
QUE LE" :: DISPLAY AT(16,2):"JOUEUR DETE
RMINE SOIT EN"
235 DISPLAY AT(18,1):"APPUYANT SUR LES T
OUCHES E," :: DISPLAY AT(20,1):"X,D OU S
, SOIT EN UTILISANT"
240 DISPLAY AT(22,5)BEEP:"LES MANETTES D
E JEU."
245 DE=500 :: GOSUB 755
250 DISPLAY AT(10,1)ERASE ALL:"EN COURS
DE JEU, POUR FAIRE" :: DISPLAY AT(12,1):
"REPETER LA DERNIERE SEQUENCE"
255 DISPLAY AT(14,1):"IL SUFFIT D'APPUYE
R SUR AID" :: DISPLAY AT(16,1):"OU D'ACT
IONNER LE BOUTON DE" :: DISPLAY AT(18,7)
BEEP:"TIR DES MANETTES."
260 DE=250 :: GOSUB 755
265 GOTO 95
270 REM ====================
275 REM Affichage de l'écran
  du jeu
280 REM ====================
285 CALL CLEAR
290 FOR I=9 TO 12 :: CALL COLOR(I,1,1)::
 NEXT I
295 FOR I=0 TO 4
300 CALL HCHAR(1+I,15-I,96,2*I+5):: CALL
 VCHAR(10-I,28-I,104,2*I+5):: CALL HCHAR
(23-I,15-I,112,2*I+5):: CALL VCHAR(10-I,
6+I,120,2*I+5)
305 NEXT I
310 FOR I=0 TO 3
315 CALL HCHAR(1+I,14-I,97):: CALL HCHAR
(1+I,20+I,98):: CALL HCHAR(6+I,25+I,106)
:: CALL HCHAR(15+I,28-I,107)
320 CALL HCHAR(20+I,23-I,115):: CALL HCH
AR(20+I,11+I,116):: CALL HCHAR(15+I,6+I,
124):: CALL HCHAR(6+I,9-I,121)
325 NEXT I
330 FOR I=0 TO 2
335 CALL HCHAR(5+I,11+I,100):: CALL HCHA
R(5+I,23-I,99):: CALL HCHAR(6+I,24-I,105
):: CALL HCHAR(16+I,22+I,108)
340 CALL HCHAR(17+I,21+I,114):: CALL HCH
AR(17+I,13-I,113):: CALL HCHAR(6+I,10+I,
122):: CALL HCHAR(18-I,10+I,123)
345 NEXT I
350 FOR I=0 TO 1
355 CALL HCHAR(6,7*I+13,96,2):: CALL HCH
AR(7-I,14+I,99):: CALL HCHAR(6+I,19+I,10
0):: CALL VCHAR(7*I+8,23,104,2)
360 CALL HCHAR(9+I,22+I,108):: CALL HCHA
R(14+I,23-I,105):: CALL HCHAR(18,7*I+13,
112,2):: CALL HCHAR(17+I,14+I,114)
365 CALL HCHAR(17+I,20-I,113):: CALL VCH
AR(7*I+8,11,120,2):: CALL HCHAR(9+I,12-I
,123):: CALL HCHAR(14+I,11+I,122)
370 NEXT I
375 GOSUB 730
380 REM ====================
385 REM  Début de Partie
390 REM ====================
395 DISPLAY AT(10,12)SIZE(7):"APPUYEZ" :
: DISPLAY AT(12,14)SIZE(3):"SUR" :: DISP
LAY AT(14,13)BEEP SIZE(5):"BEGIN"
400 DE=300 :: GOSUB 755
405 IF K=7 OR K=51 THEN GOSUB 1005 ELSE
415
410 GOTO 285
415 IF KT=18 OR K=14 OR K=53 THEN GOSUB
795 ELSE 95
420 REM ====================
425 REM          Jeu
430 REM ====================
435 M=1 :: N=1
```

```
440 D=500-INT(104*LOG(N)):: DE=100+INT(2
00/N)
445 SC=Z+10*(N-1)
450 CALL HCHAR(1,30,48):: CALL HCHAR(1,2
9,48+N-100*INT(N/100)-10*INT(N/10))
455 CALL HCHAR(1,28,48+INT(N/10)-10*INT(
N/100)):: CALL HCHAR(1,27,48+INT(Z/1000)
)
460 IF KM=50 THEN 490
465 M=1
470 RANDOMIZE :: P(2,N)=INT(RND*4)+1
475 GOSUB 820
480 CALL HCHAR(12,17,43)
485 GOTO 495
490 CALL HCHAR(12,17,48+M)
495 FOR O=1 TO N
500 A=0
505 CALL KEY(3,K,S):: CALL KEY(M,KT,S)::
 CALL JOYST(M,X,Y)
510 IF KT=18 OR K=1 OR K=55 THEN GOSUB 8
20 :: GOTO 495
515 IF K=69 OR Y=4 THEN P(M,O)=1 :: GOTO
 550
520 IF K=68 OR X=4 THEN P(M,O)=2 :: GOTO
 550
525 IF K=88 OR Y=-4 THEN P(M,O)=3 :: GOT
O 550
530 IF K=83 OR X=-4 THEN P(M,O)=4 :: GOT
O 550
535 A=A+1
540 IF A=DE THEN 615
545 GOTO 505
550 IF P(2/M,O)=0 THEN 560
555 IF P(M,O)<>P(2/M,O)THEN 615
560 ON P(M,O)GOSUB 845,885,910,965
565 NEXT O
570 GOSUB 795
575 M=2/M
580 N=N+1
585 IF N<100 THEN 440
590 Z=Z+1000
595 GOTO 625
600 REM ====================
605 REM   Fin de Partie
610 REM ====================
615 GOSUB 795
620 DISPLAY AT(12,11)SIZE(9):"GAME OVER"
625 D=30
630 FOR I=1 TO N :: ON 1+P(2/M,I)GOSUB 8
65,845,885,925,965 :: P(1,I)=0 :: P(2,I)
=0 :: NEXT I
635 IF N=100 THEN 285
640 IF SC>B(5)THEN 670
645 DISPLAY AT(24,4)BEEP:"APPUYEZ SUR RE
DO OU BACK"
650 DE=200 :: GOSUB 755
655 IF K=15 OR K=57 THEN 95 ELSE 285
660 REM ====================
665 REM  Meilleurs Scores
670 REM ====================
675 FOR I=1 TO 4
680 IF SC>B(I)THEN 690
685 NEXT I
690 FOR J=5 TO I+1 STEP -1 :: B$(J)=B$(J
-1):: B(J)=B(J-1):: NEXT J
695 B(I)=SC
700 DISPLAY AT(15,2)ERASE ALL BEEP:"VOS
INITIALES SVP ?" :: ACCEPT AT(15,23)VALI
DATE(UALPHA)SIZE(3)BEEP:B$(I)
705 GOSUB 1005
710 GOTO 645
715 REM ====================
720 REM   Colorisation
725 REM ====================
730 CALL COLOR(9,7,1):: CALL COLOR(10,11
,1):: CALL COLOR(11,5,1):: CALL COLOR(12
,13,1)
735 RETURN
740 REM ====================
745 REM      Call Key
750 REM ====================
755 A=0
760 CALL KEY(3,K,S):: CALL KEY(1,KT,ST):
: A=A+1
765 IF A=DE THEN 775
770 IF S=0 AND KT<>18 THEN 760
775 RETURN
780 REM ====================
785 REM  Effacage Central
790 REM ====================
795 CALL HCHAR(10,14,32,9):: CALL HCHAR(
12,13,32,7):: CALL HCHAR(14,15,32,5)
800 RETURN
805 REM ====================
810 REM <Aid>  Répétition
815 REM ====================
820 FOR I=1 TO N :: ON 1+P(2/M,I)GOSUB 8
25,845,885,925,965 :: NEXT I
825 RETURN
830 REM ====================
835 REM  Rouge <349Hz>
840 REM ====================
845 CALL COLOR(9,1,1)
850 CALL SOUND(D,349,0)
855 CALL SOUND(D,44730,30)
860 CALL COLOR(9,7,1)
865 RETURN
870 REM ====================
875 REM  Jaune <440Hz>
880 REM ====================
885 CALL COLOR(10,1,1)
890 CALL SOUND(D,440,0)
895 CALL SOUND(D,44730,30)
900 CALL COLOR(10,11,1)
905 RETURN
910 REM ====================
915 REM  Bleu  <587Hz>
920 REM ====================
```

```
925 CALL COLOR(11,1,1)
930 CALL SOUND(D,587,0)
935 CALL SOUND(D,44730,30)
940 CALL COLOR(11,5,1)
945 RETURN
950 REM ===================
955 REM  Vert  <659Hz>
960 REM ===================
965 CALL COLOR(12,1,1)
970 CALL SOUND(D,659,0)
975 CALL SOUND(D,44730,30)
980 CALL COLOR(12,13,1)
985 RETURN
990 REM ===================
995 REM       Affichage
des  Scores
1000 REM ===================
1005 DISPLAY AT(1,3)ERASE ALL BEEP:"SCOR
ES :"
1010 FOR I=1 TO 5 :: DISPLAY AT(3*I+5,13
):B$(I):: DISPLAY AT(3*I+5,17):B(I):: NE
XT I
1015 DE=300 :: GOSUB 755
1020 RETURN
1025 REM ===================
1030 REM        Data
1035 REM ===================
1040 DATA 2,11,98,2,10,96,2,9,96,2,8,97,
3,8,96,4,8,100,4,9,96,4,10,96
1045 DATA 4,11,98,5,11,96,6,11,99,6,10,9
6,6,9,96,6,8,100
1050 DATA 4,13,104,5,13,104,6,13,104
1055 DATA 6,15,112,5,15,112,4,15,112
1060 DATA 4,16,112,4,17,112,5,17,112,6,1
7,112,4,18,112,4,19,114,5,19,112,6,19,11
2
1065 DATA 4,22,120,4,23,122,5,23,120,6,2
3,123,6,22,120,6,21,124,5,21,120,4,21,12
1
1070 DATA 6,25,96,5,25,96,4,25,96,4,26,9
6,4,27,98,5,27,96,6,27,96,2,13,105
```

Jeu des carrés

Didier Bicking

Voici encore un programme qui nous vient de nos amis belges du club TI-SOFT-HOME, décidément très prolifique. Mais cette fois-ci, faute de temps, je ne l'ai pas traduit en français. Il vous appartient donc de faire ce travail. En fait, seuls les messages doivent être modifiés et je vous conseille, lorsque vous taperez ce programme, de repérer sur le listing les ordres PRINT devant supporter cette modification. Vous taperez alors directement les textes en français.

Le principe de ce jeu est très connu : tout le monde y a joué un jour ou l'autre, utilisant pour ce faire une page d'un cahier scolaire. Chaque joueur à son tour trace un trait horizontal ou vertical. Le jeu consiste à être celui qui fermera le plus de carrés dans la surface délimitée pour le jeu. Ainsi, quand un carré comporte déjà trois côtés, il faut tracer le quatrième pour que le carré soit pris.

Le programme offre trois dimensions de jeu. Cela ne signifie pas qu'il y a trois niveaux de stratégie : l'ordinateur ne se comporte ici que comme une feuille de papier, enregistrant les coups des adversaires. Mais il fait cela très joliment : les représentations graphiques sont très soignées et les illustrations sonores très astucieuses.

Ainsi, à chaque joueur correspond une couleur de fond d'écran ainsi qu'une musique l'invitant à jouer. Lorsque l'un des joueurs ferme un carré, une petite musique l'en félicitent et le carré est coloré dans la couleur du joueur.

Il n'y a aucune astuce particulière à signaler dans ce programme. C'est du beau travail, bien propre et bien fait. Notons surtout la présence de REMs en nombre suffisant : ils contribuent à rendre le listing compréhensible, ce qui est tout aussi agréable pour le programmeur qui met sont travail au point, que pour les lecteurs de "99", non ?

```
50 REM ****************************
60 REM *  COPYRIGHT 99 MAGAZINE    *
70 REM *      ET TI-SOFT HOME      *
80 REM ****************************
90 REM
100 RANDOMIZE
110 CALL CLEAR
120 GOSUB 2680
130 FOR X=1 TO 8
140 CALL COLOR(X,2,4)
150 NEXT X
160 L=12
170 C=16
180 PRINT "      NOMBRE DE JOUEURS? "
190 CALL KEY(0,KY,ST)
200 IF ST=0 THEN 190
210 NO=VAL(CHR$(KY))
220 IF NO<5 THEN 250
230 PRINT " PAS PLUS DE QUATRE JOUEURS":
 :
240 GOTO 180
250 PRINT " DIMENSION DU JEU (P,M,G)"
260 OCHAR=96
270 PL=1
280 CALL KEY(0,KY,ST)
290 IF ST<>1 THEN 280
300 IF KY<>80 THEN 370
310 TOP=9
320 BOT=16
330 LEFT=12
340 RIGHT=19
350 PLAYS=60
360 GOTO 500
370 IF KY<>77 THEN 440
```

```
380 TOP=5
390 BOT=19
400 LEFT=9
410 RIGHT=23
420 PLAYS=312
430 GOTO 500
440 IF KY<>71 THEN 280
450 TOP=2
460 BOT=23
470 LEFT=6
480 RIGHT=27
490 PLAYS=760
500 REM
510 CALL CHAR(96,"01")
520 FOR A=0 TO 8 STEP 8
530 CALL CHAR(104+A,"FF")
540 CALL CHAR(105+A,"0101010101010101")
550 CALL CHAR(106+A,"FF01010101010101")
560 CALL CHAR(107+A,"FF818181818181FF")
570 NEXT A
580 FOR Z=120 TO 152 STEP 8
590 CALL CHAR(Z,"FF818181818181FF")
600 NEXT Z
610 CALL COLOR(1,4,4)
620 CALL COLOR(9,2,4)
630 CALL COLOR(10,2,4)
640 CALL COLOR(11,16,4)
650 CALL COLOR(12,2,5)
660 CALL COLOR(13,2,7)
670 CALL COLOR(14,2,10)
680 CALL COLOR(15,2,13)
690 CALL HCHAR(1,1,96,768)
700 FOR Z=1 TO TOP
710 CALL HCHAR(Z,1,32,32)
720 NEXT Z
730 FOR Z=BOT TO 24
740 CALL HCHAR(Z,1,32,32)
750 NEXT Z
760 FOR Z=1 TO LEFT
770 CALL VCHAR(1,Z,32,24)
780 NEXT Z
790 FOR Z=RIGHT TO 32
800 CALL VCHAR(1,Z,32,72)
810 NEXT Z
820 CALL HCHAR(L,C,104+INT(RND*2+1)-1)
830 CALL SCREEN(5)
840 CALL HCHAR(L,C,104)
850 BLOC=104
860 NCHAR=104
870 GOTO 1580
880 REM *****************************
890 REM *  ROUTINE DU CURSEUR       *
900 REM *****************************
910 CALL KEY(0,KY,ST)
920 IF ST=0 THEN 910
930 IF KY=13 THEN 1530
940 CALL HCHAR(L,C,OCHAR)
950 CALL SOUND(100,1000*RND+110,15)
960 IF KY<>69 THEN 1050
970 IF C<>LEFT+1 THEN 990
980 C=LEFT+2
990 BLOC=104
1000 L=L-1
1010 IF L>TOP THEN 1030
1020 L=BOT-1
1030 CALL GCHAR(L,C,OCHAR)
1040 GOTO 1370
1050 IF KY<>88 THEN 1140
1060 IF C<>LEFT+1 THEN 1080
1070 C=LEFT+2
1080 BLOC=104
1090 L=L+1
1100 IF L<BOT THEN 1120
1110 L=TOP+1
1120 CALL GCHAR(L,C,OCHAR)
1130 GOTO 1370
1140 IF KY<>68 THEN 1230
1150 IF L<>BOT-1 THEN 1170
1160 L=BOT-2
1170 BLOC=105
1180 C=C+1
1190 IF C<RIGHT THEN 1210
1200 C=LEFT+1
1210 CALL GCHAR(L,C,OCHAR)
1220 GOTO 1310
1230 IF KY<>83 THEN 910
1240 IF L<>BOT-1 THEN 1260
1250 L=BOT-2
1260 BLOC=105
1270 C=C-1
1280 IF C>LEFT THEN 1300
1290 C=RIGHT-1
1300 CALL GCHAR(L,C,OCHAR)
1310 REM BLOC=105
1320 IF OCHAR<104 THEN 1450
1330 IF OCHAR=104 THEN 1480
1340 IF OCHAR=105 THEN 1450
1350 IF OCHAR=106 THEN 1480
1360 IF OCHAR>106 THEN 1510
1370 REM BLOC=104
1380 IF OCHAR=105 THEN 1480
1390 IF OCHAR=106 THEN 1480
1400 IF OCHAR>106 THEN 1510
1410 IF OCHAR<=104 THEN 1420
1420 CALL HCHAR(L,C,112)
1430 NCHAR=104
1440 GOTO 910
1450 CALL HCHAR(L,C,113)
1460 NCHAR=105
1470 GOTO 910
1480 CALL HCHAR(L,C,114)
1490 NCHAR=106
1500 GOTO 910
1510 CALL HCHAR(L,C,115)
1520 GOTO 910
1530 RETURN
1540 PLAYS=PLAYS-1
1550 IF PLAYS=0 THEN 2510
```

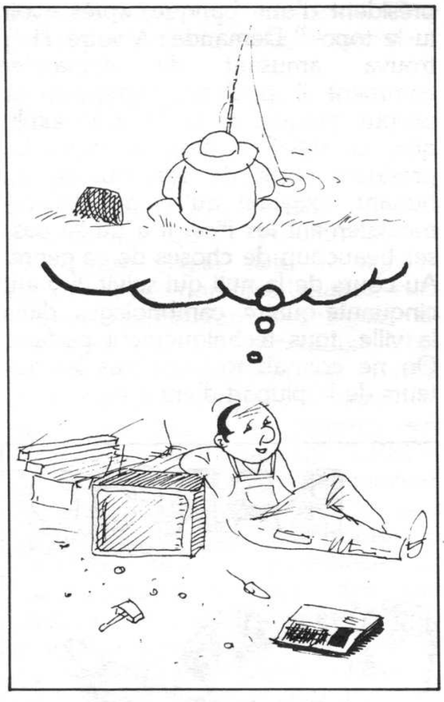
```
1560 PR$=STR$(PLAYS)&" "
1570 GOSUB 2640
1580 GOSUB 910
1590 IF OCHAR>105 THEN 1610
1600 IF BLOC<>OCHAR THEN 1630
1610 CALL SOUND(300,110,0)
1620 GOTO 1580
1630 CALL GCHAR(L-1,C,TST1)
1640 CALL GCHAR(L-1,C-1,TST2)
1650 CALL GCHAR(L+1,C,TST3)
1660 CALL GCHAR(L,C+1,TST4)
1670 CALL GCHAR(L,C-1,TST5)
1680 CALL GCHAR(L+1,C+1,TST6)
1690 IF BLOC=105 THEN 1920
1700 REM ****************************
1710 REM *          EN HAUT         *
1720 REM ****************************
1730 IF TST1<106 THEN 1800
1740 IF TST2<105 THEN 1800
1750 CALL HCHAR(L-1,C,112+PL*8)
1760 SKIP=1
1770 SCORE(PL)=SCORE(PL)+1
1780 GOSUB 2680
1790 REM ****************************
1800 REM *          EN BAS          *
1810 REM ****************************
1820 IF TST3=104 THEN 1840
1830 IF TST3<106 THEN 2170
1840 IF TST5<105 THEN 2170
1850 IF OCHAR<>105 THEN 2170
1860 CALL HCHAR(L,C,112+PL*8)
1870 SKIP=0
1880 SCORE(PL)=SCORE(PL)+1
1890 GOSUB 2680
1900 GOTO 2130
1910 REM ****************************
1920 REM *     A GAUCHE             *
1930 REM ****************************
1940 IF TST4<>106 THEN 2040
1950 IF TST6=104 THEN 1970
1960 IF TST6<106 THEN 2040
1970 CALL HCHAR(L,C+1,112+PL*8)
1980 SKIP=1
1990 SCORE(PL)=SCORE(PL)+1
2000 GOSUB 2680
2010 REM ****************************
2020 REM *          A DROITE        *
2030 REM ****************************
2040 IF OCHAR<>104 THEN 2170
2050 IF TST3=104 THEN 2070
2060 IF TST3<106 THEN 2170
2070 IF TST5<105 THEN 2170
2080 CALL HCHAR(L,C,112+PL*8)
2090 SKIP=0
2100 SCORE(PL)=SCORE(PL)+1
2110 GOSUB 2680
2120 REM
2130 CALL GCHAR(L,C,TEMP)
2140 OCHAR=TEMP
2150 CALL HCHAR(L,C,OCHAR)
2160 GOTO 1540
2170 OCHAR=NCHAR
2180 CALL HCHAR(L,C,NCHAR)
2190 IF SKIP<>1 THEN 2230
2200 SKIP=0
2210 GOTO 1540
2220 REM ****************************
2230 REM *  SELECTION DU JOUEUR     *
2240 REM *        SUIVANT           *
2250 REM ****************************
2260 PL=PL+1
2270 IF PL>NO THEN 2450
2280 IF PL=2 THEN 2350
2290 IF PL=3 THEN 2400
2300 CALL SCREEN(13)
2310 CALL SOUND(100,392,0)
2320 CALL SOUND(100,262,0)
2330 CALL SOUND(100,330,0)
2340 GOTO 1540
2350 CALL SCREEN(7)
2360 CALL SOUND(100,262,0)
2370 CALL SOUND(100,392,0)
2380 CALL SOUND(100,330,0)
2390 GOTO 1540
2400 CALL SCREEN(10)
2410 CALL SOUND(100,392,0)
2420 CALL SOUND(100,330,0)
2430 CALL SOUND(100,262,0)
2440 GOTO 1540
2450 PL=1
2460 CALL SCREEN(5)
2470 CALL SOUND(100,262,0)
2480 CALL SOUND(100,330,0)
2490 CALL SOUND(100,392,0)
2500 GOTO 1540
2510 PRINT " LE JEU EST TERMINE"
2520 PRINT :"SCORES FINAUX:"
2530 FOR X=1 TO NO
2540 PRINT "  -";SCORE(X)
2550 CALL HCHAR(23,4,112+X*8)
2560 NEXT X
2570 FOR DELAY=1 TO 1000
2580 NEXT DELAY
2590 PRINT " VOULEZ-VOUS REJOUER (O/N)?"
2600 CALL KEY(0,KY,ST)
2610 IF ST=0 THEN 2600
2620 IF KY=79 THEN 250
2630 END
2640 FOR X=1 TO LEN(PR$)
2650 CALL HCHAR(24,X+3,ASC(SEG$(PR$,X,1)
))
2660 NEXT X
2670 RETURN
2680 CALL SOUND(300,262,0,330,0,392,0)
2690 CALL SOUND(500,262,0,392,0,524,0)
2700 CALL SOUND(700,1047,0,524,0,262,0)
2710 RETURN
2720 BLOC=104
```

L'ordinateur de l'avenir

Didier Bicking

Cela faisait maintenant sept ans, ce 23 mai 1983, que j'avais épousé Sylvie; et c'est cette après-midi que choisit Martine pour débarquer dans ma ville. Cette après-midi là, j'ai sauvé la civilisation humaine. C'est du moins comme cela que je vois les choses. Martine est une blonde dont j'étais cinglé dans le temps.

Le rôle que Martine a joué là-dedans, c'est qu'elle me faisait passer des frissons glacés le long de la colonne vertébrale, rien que de penser à elle. En fait, j'ai une femme que j'ai épousée après m'être séparé de Martine avec un désespoir du genre romantique. C'est une bonne épouse et elle m'a donné des gosses. Ils sont insupportables, mais ce sont les miens et je les aime bien. Si je me tiens tranquille, je pourrai finir ma vie à pêcher à la ligne, avec une pension de la Sécurité Sociale et quelques économies. Oui mais il y a Joe, et il m'inquiète Joe !

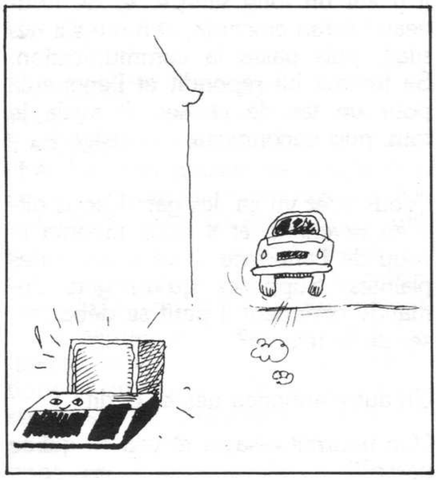


Je travaille pour une société qui fabrique des terminaux intelligents. Mon rôle consiste à réparer les machines en panne et, en toute modestie, je dois dire que je ne me débrouille pas mal. Avant, je réparais des téléviseurs. Jusqu'au jour où on a inventé ce système qui consistait à brancher des micro-ordinateurs sur une banque de données générale. Depuis, d'ailleurs, on ne parle plus de micro-ordinateurs mais seulement de terminaux intelligents (on dit TI tout court). Les gens qui ont conçu ce système ont commencé par brancher entre eux plusieurs ordinateurs, puis ces branchements se sont généralisés et ils étaient contents. Ils en sont toujours à se demander tout ce que l'on peut faire avec un TI mais, en attendant, tout le monde en a un chez soi. Moi-même, j'ai Joe, mais il est à la cave et je ne sais toujours pas si je vais le remettre en marche ou le mettre en miettes avant de le jeter au plus profond de la rivière.

Vous voyez le tableau. Vous avez un TI chez vous. Ca ressemble à un poste de télévision sauf qu'il y a un clavier au lieu de boutons; vous y tapez ce que vous voulez obtenir. Il est relié à la banque de données générale. Mettons que vous tapiez "Station TF1" et la banque vous envoie le programme diffusé par TF1. Ou bien vous tapez "Téléphone de Jacqueline Durand", et vous vous trouvez relié, son et image, avec le TI de Jacqueline Durand. Et si vous demandez la météo, ou le résultat du tiercé aujourd'hui, ou qui était ministre des finances sous Pompidou, vous l'aurez aussi sur l'écran. A cause des fichiers de la banque. La banque, c'est un grand bâtiment qui contient tous les faits de la création et des enregistrements de toutes les émissions jamais réalisées. Elle contient aussi des tas de programmes que peut utiliser votre TI pour vous faire tous les travaux ou tous les jeux que vous pouvez imaginer. Elle est reliée à toutes les autres banques du pays et tout ce que vous voulez savoir, entendre ou faire, vous tapez et ça vient. Ca tient le rôle de médecin, de pharmacien, physicien, astronome et diseuse de bonne aventure. La seule chose que le TI ne vous dira jamais, c'est ce que votre femme veut dire au juste quand elle déclare de cette voix si particulière : "Ah ! Tu crois ça ?" Les TI ne sont pas très forts en ce qui concerne les femmes. Il faut qu'ils aient affaire à des choses sensées.

On nous a dit que les TI ont changé la civilisation. Et Joe aurait dû être un TI parfaitement normal, faisant à la place de pauvres parents les devoirs des enfants. Mais il a dû y avoir une erreur en cours de montage. Une erreur si petite que les instruments de précision ne l'ont pas décelée, mais qui fit que Joe est un individu. Au début, il ne le savait pas lui-même ou bien, étant un être logique, il se disait que, si l'on réalisait qu'il était différent, on le mettrait à la ferraille. Soit dit en passant, ç'aurait été une idée brillante. Bref, il passa tous les tests sans qu'on s'aperçoive de rien, et il fut installé chez un client. Jusque là, beau fixe.

L'installation eut lieu dans la soirée du samedi. Le dimanche matin, les enfants de la famille regardèrent les émissions pour enfants de la semaine passée. A midi, les parents arrachèrent les enfants au TI et les empilèrent dans la voiture. Puis ils revinrent prendre les sandwiches qu'ils avaient oubliés, et un des gosses en profita pour demander encore une émission enfantine. Ses parents le firent sortir dare-dare, mais ils oublièrent d'éteindre Joe.



Il était midi. Rien ne se passa avant deux heures. C'était le calme qui précède la tempête. Martine n'était pas encore arrivée en ville, mais cela n'allait pas tarder. J'imagine Joe, tout seul dans l'appartement, ronronnant et méditatif. Pendant un bout de temps, il a dû passer des émissions enfantines pour le plus grand bénéfice du buffet de la salle à manger, mais je crois qu'il a fini par aller explorer la banque. Il n'y a pas un fait digne de ce nom qui ne soit pas quelque part dans une de ces banques. Ce n'est pas la matière première qui manquait. Et Joe se mit au boulot.

Non qu'il soit vicieux, vous comprenez. Il n'est pas comme un de ces méchants robots qui veulent remplacer la race humaine par des machines pensantes. Mais Joe était ambitieux. Si vous étiez une machine, vous voudriez faire votre boulot le mieux possible, non ? Eh bien, Joe,

c'est ça. C'est un TI et il veut bien faire son travail. Et les TI peuvent faire bien des choses que les humains ne peuvent pas faire, ou auxquelles ils n'ont pas encore pensé, et il essaie de faire en sorte qu'on demande aux TI de les faire. C'est tout. Absolument tout, mais ça suffit, croyez-moi !

Vers les deux heures de l'après-midi, les choses sont plutôt calmes au service entretien dans lequel je travaille. On jouait aux cartes. Puis un des gars se souvint qu'il devait appeler sa femme. Il alla vers l'un des TI et demanda sa maison. L'écran crachota puis s'illumina.

"Nous avons le plaisir de vous annoncer une extension et une amélioration de nos services ! Votre TI peut dès maintenant, en plus des informations, vous donner des conseils ! Si vous voulez faire quelque chose et ne savez pas comment vous y prendre, demandez-le au TI !"

Il y eut un long silence, et de nouveau l'écran crachota, comme s'il hésitait, puis passa la communication. Sa femme lui répondit et l'engueula pour un tas de choses. Il avala le tout, puis déconnecta.

"Vous avez vu ça, les gars ! nous dit-il en revenant, et il nous raconta le coup de l'annonce. Il va y avoir des plaintes. Supposez qu'un gars demande comment il peut se débarrasser de sa femme?"

Un autre annonça des as et dit :

"On pourrait essayer et voir ce qui se passe?"

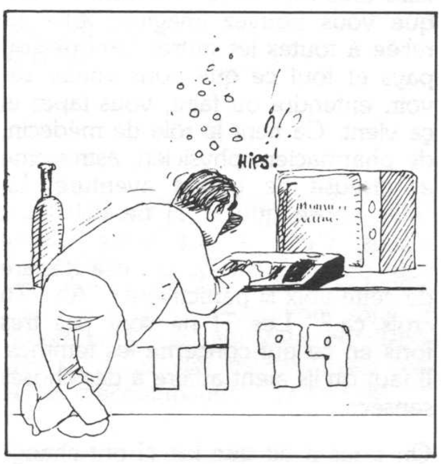
Pour rire, bien sûr. Il y alla et tapa sa question. En principe, dans ce genre de cas, une cloche de la Censure apparaissait sur l'écran et disait d'une voix sévère : "La Sécurité Publique nous interdit de répondre à votre question." Il faut bien une censure, sans ça les gosses demanderaient des choses qu'ils sont trop jeunes pour apprendre. Et pour d'autres raisons, comme vous allez le voir.

Le gars tapa sa question : "Comment faire pour me débarrasser de ma femme ?" Il en rigolait d'avance. L'écran resta vide une demi-seconde puis : "Question de service : votre femme est-elle blonde, rousse ou brune?" Du coup il rigola franchement et nous fit signe d'approcher. Il tapa : "Blonde". Une autre pause d'une demi-seconde, puis l'écran afficha : "Le cirage pour chaussures de couleur verte contient de l'hexamétacryloaminoacétine. Préparez un repas contenant de la soupe aux pois secs. Colorez-la avec du cirage vert. Elle aura l'aspect d'une soupe de pois frais. L'hexamétacryloaminoacétine est un poison fatal pour les femmes blondes, mais il est sans effet sur les autres femmes, ou sur les hommes quelle que soit la couleur de leurs cheveux. Ce fait n'a pas été découvert par des expérimentateurs humains, mais est un produit de nos services. On ne pourra pas vous inculper de meurtre. Il est même fort improbable que vous soyez jamais suspecté."

L'écran redevint vide alors que nous nous regardions avec stupéfaction. Il devait avoir raison : un TI ne fait pas plus d'erreur qu'une simple calculette électronique. J'appelai les gars de la banque centrale sans perdre un instant :

"Hé, les gars ! Il se passe quelque chose ! Les TI se mettent à donner des instructions détaillées pour tuer sa femme ! Vite ! Vérifiez les programmes de censure !"

C'avait été de justesse ! Mais j'étais naïf. Au même instant, dans la Grand-Rue, un ivrogne tapa quelque chose sur un TI public. L'écran lui fit son topo sur les nouveaux services. "Chiche !" dit l'ivrogne et il composa une autre question : "Comment empêcher ma femme de s'apercevoir que j'ai bu?" Et la réponse vint : "Achetez un flacon de shampooing Phagor. Ce liquide inoffensif contient un détergent qui neutralise rapidement l'alcool éthylique. Dosage à utiliser : une cuiller à café pour chaque gramme d'alcool que vous avez bu."



Le gars en question était suffisamment rond pour suivre ce conseil. Cinq minutes après sa sortie de la pharmacie, il était dessoûlé. Suffisamment pour voir le parti qu'il pouvait en tirer. Il fit breveter PASSOU, *la boisson qui rendra votre foyer heureux*. Son seul problème maintenant, c'est l'impôt sur le revenu !

Bon ! Ça ce n'est pas grave. Mais voilà un gosse de quatorze ans qui voulait s'acheter une bicyclette, et son papa ne voulait pas débourser. Il appella un copain pour lui raconter ses ennuis et son TI lui dit : " si vous voulez quelque chose sans savoir comment, demandez à votre TI." Alors le gosse composa : " Comment gagner très vite un tas d'argent ? "

Son TI lui fournit le moyen le plus simple et le plus parfait de contrefaire les billets de banque jamais inventé. Voyez-vous, toutes les données de base étaient dans la banque. Le TI n'avait qu'à intégrer les faits – chose possible depuis que Joe avait neutralisé certains programmes ça et là dans la banque. C'était tout. Le gosse se fit prendre au bout de trois jours ; il avait dépensé des fortunes et avait encore de l'argent plein les poches. Il était presque impossible de distinguer ses billets des vrais. Il ne s'était d'ailleurs fait prendre que parce qu'il avait essayé d'améliorer le procédé, comme les gosses qui ne sont jamais contents tant que ça marche.

Ce ne sont là que des exemples. Personne ne sait au juste tout ce que Joe a fait. Il y a aussi le jour où le président d'une banque, après avoir lu le topo " Demandez à votre TI ", trouva amusant de demander comment il pourrait cambrioler sa propre banque. Et le TI le lui expliqua, en détail et pour de bon ! Le président sortit de son bureau en hurlant, exigeant qu'on appelle immédiatement les flics. Il a dû se passer beaucoup de choses de ce genre. Au cours de la nuit qui suivit, il y eut cinquante-quatre cambriolages dans la ville, tous techniquement parfaits. On ne connaît toujours pas les auteurs de la plupart d'entre eux.



Joe avait simplement exploré la banque et neutralisé quelques programmes, comme un TI doit savoir le faire

(mais seulement sur demande !) ; en fait, il avait bloqué tous les programmes de censure et monté son " service ". Les TI concevaient alors des crimes, des machines à contrefaire, des repas agréables et nutritifs, de nouvelles industries, le tout avec une totale impartialité. Il devait être sacrément content, Joe. Il marchait à plein tandis que ses jeunes propriétaires roulaient en voiture avec papa et maman.

Ils revinrent à sept heures pile – les gosses tout heureux d'avoir fait les zouaves sur le siège arrière – puis les parents virent l'écran qui clignotait méditativement, passant d'un sujet à l'autre. Le père, qui en avait assez pour sa journée, l'éteignit.

Et la paix descendit sur terre ! Pour tout le monde. Mais pas pour moi ! Martine était arrivée en ville. J'ai souvent remercié Dieu qu'elle ne m'ait pas épousé quand je le désirais. Au début, elle était blonde et fatale puis, au fil des années, elle est devenue de plus en plus blonde et fatale et elle a eu quatre maris de suite plus un acquittement à l'issue d'un procès pour homicide ! Elle est devenue de plus en plus enthousiaste et confiante dans ses possibilités. Ceci pour dire que Martine n'était pas le genre d'expetite amie que l'on aime voir arriver dans la ville où l'on vit avec sa femme. Mais elle arriva, juste au début de la seconde crise d'activité de Joe.

Les enfants l'avaient remis en marche. J'ai appris tous ces détails par la suite, vous comprenez. Et tous les TI de la ville de répéter le petit topo : "Si vous voulez faire quelque chose, demandez-nous comment." Encore mieux : en même temps que les informations, ils donnaient un récit détaillé de tous les évènements de la veille. Cela donnait aux gens l'envie de profiter de la rigolade. Un malin demanda : "Comment puis-je fabriquer un mouvement perpétuel?" Après quelques hésitations, le TI pondit un truc utilisant les mouvements browniens. Si les rouages n'ont pas plus de trois millimètres de diamètre, ils tournent, et c'est pratiquement le mouvement perpétuel. Un autre demanda le secret de la transmutation des métaux. Le TI mit un bon bout de temps pour intégrer toutes les données connues sur ce sujet, puis donna une réponse parfaitement utilisable. Cela exige tellement d'énergie que ce n'est guère rentable que pour le radium, mais ce n'est déjà pas si mal. Et, vu que la police n'a cessé pendant deux ans de découvrir des pinces-monseigneur améliorées, des engins pour éventrer les coffres-forts et des clés universelles pour ouvrir toutes les serrures, il faut croire qu'il y en avait qui avaient l'esprit pratique. C'est fou ce que Joe a pu faire pour le progrès !

Et dans le domaine éducatif ! Mes gosses sont trop petits pour s'intéresser à ces choses, mais Joe avait bloqué tous les programmes de censure parce qu'ils empêchaient les TI de bien servir l'humanité. Alors les gosses qui voulaient savoir ce qui se passe après le stade des abeilles et des petites fleurs l'apprirent. Et il y a aussi des trucs que les gens ne tiennent pas tellement à dévoiler à leurs conjoints, mais ce sont justement ceux-là qui les intéressent. Alors, quand une femme demande : "Comment savoir si Jean-Jacques m'est fidèle ?" et que son TI le lui dit... il y a de la bagarre en perspective !

Pendant ce temps, Joe, ronronnant avec contentement, passait des dessins animés à ses jeunes propriétaires avec un de ses circuits tandis qu'avec les autres il téléguidait la banque de façon que les autres TI puissent donner aux gens ce qu'ils désiraient, et par la même occasion faire un grabuge de tous les diables.

C'est là que Martine entra en jeu. Elle ouvrit le TI de sa chambre d'hôtel, sans doute pour demander l'émission de mode, mais le TI, bien sûr, lui passa sa petite annonce : "Si vous voulez faire quelque chose, etc." Martine, forcément, fut enthousiasmée et elle se demanda quelle question poser. Elle n'avait plus rien à apprendre-pardi, elle avait eu quatre maris dont un qu'elle avait liquidé - alors elle pensa à moi. Et elle composa : "Comment puis-je retrouver Doucky?"

Oui, c'est ainsi qu'elle m'appelait dans le temps. Le TI lui posa une question de service : "Doucky est-il connu sous autre nom?" Elle lui donna mon nom, mais le TI ne put pas me trouver parce que mon TI à moi n'est pas à mon nom. Comme je travaille pour la maison, je l'ai eu gratis.

Joe fut embarassé. C'était sans doute la première question à laquelle il ne pouvait pas répondre. Un drôle de problème. Joe y réfléchit tout en montrant aux enfants un dessin animé sur le malin petit garçon qui a les poches pleines de bâtons de dynamite et joue des tours à tous ses voisins. Puis il trouva la solution. L'écran de Martine passa un petit avis : "nous vous rappellerons dès que nous aurons résolu votre problème."

Martine commençait déjà à bailler, mais elle fit ce qu'il lui demandait et s'en alla faire la sieste. Joe, quant à lui, se mit au travail. Cela lui avait donné une idée.

Ma femme m'appela à ce moment là et se mit à hurler. Elle était folle de rage, il fallait que je fasse quelque chose ! Elle avait voulu appeler le boucher. Au lieu de lui passer le boucher, ou même le topo habituel, le TI lui avait demandé : "Question de service : comment vous appelez-vous ?" Elle, un peu étonnée, répondit. Alors l'écran crachota un moment et dit : "Démonstration de notre service documentaire." Puis il se mit à débiter nom, âge, adresse, sexe, couleur des cheveux, nom du mari, montant de son compte en banque, montant des dettes chez les commerçants, trois condamnations que j'avais reçues - deux pour des histoires de circulation et la troisième pour désordre sur la voie publique - et le détail amusant qu'une fois, mon épouse était si en colère contre moi qu'elle m'a abandonné durant trois semaines et a fait suivre le courrier chez papa et maman. Et il avait ajouté gaiement : "Ceci n'était qu'une démonstration. A l'avenir, nous pourrons prendre vos messages, rechercher des personnes que vous désirez contacter, donner tous les détails biographiques, etc. Merci pour votre attention." Puis il lui passa le boucher.

Mais elle n'avait déjà plus envie de viande. Elle voyait rouge et m'avait appelé. "Et il va dire ça au premier qui le lui demandera ! Il faut que tu fasses quelque chose." Elle ne se tenait plus de rage.

"Allons, allons, chérie." J'essayai de la calmer. Je dis pour la rassurer " Je n'en savais rien, mais je suis sûr qu'ils ne donneront ces renseignements qu'à la personne elle-même. "

- Tu parles ! explosa-t-elle. J'ai essayé ! Tu connais notre voisine Madame Henry ? Eh bien, elle a été mariée trois fois, elle a quarante deux ans - elle dit qu'elle n'en a que trente cinq ! - et son mari a été arrêté quatre fois pour abandon de famille et une fois pour coups et blessures. Et...

- Et c'est le TI qui t'a dit tout ça ?

- Oui, hurla-t-elle. Il dit n'importe quoi à n'importe qui ! Il faut faire cesser ça tout de suite. Ca sera long ?

- Je vais appeler la banque centrale, dis-je. Ca devrait être vite réparé.

- Dépêche-toi, me dit-elle avec désespoir, avant que quelqu'un ne

compose mon nom... Je vais voir ce qu'il dit sur l'idiote d'en face."

Et elle raccrocha pour en profiter avant que ça cesse. Moi je demandai la banque et reçus l'avis : "Comment vous appelez-vous ?" Pris d'une curiosité morbide, je répondis et le TI me demanda : "Avez-vous été surnommé Doucky?" Etonné mais sans méfiance, je répondis "Oui." Et l'écran me dit : "Il y a un appel pour vous."

Et pan dans le mille ! Je vis l'intérieur d'une chambre d'hôtel avec Martine endormie sur le lit. Il faisait chaud ce jour là mais elle avait fait de son mieux pour ne pas souffrir de la chaleur. Moi, je suis humain, je ne restai pas de glace. Inutile d'insister. Je repris mon souffle et m'écriai : "Nom de Dieu !" Elle ouvrit les yeux.

Au début, elle parut surprise de me voir sur l'écran, comme si elle se demandait si je n'étais pas un des gars qu'elle avait épousés récemment. Puis elle s'enroula dans un drap et me lança un sourire radieux. "Doucky, c'est formidable !"

J'étais couvert de sueur. Je marmonnai quelque chose dans le genre de "Umph !"

- Je me sens si seule ! Depêche-toi, Doucky. Je ferai monter quelque chose à boire. Tu penses souvent à moi, Doucky ?

- Oui, articulai-je faiblement. Très souvent.

- Tu es un amour, me dit-elle en m'envoyant un baiser. Tiens !, c'est un acompte. Je t'attends, mon Doucky !"

Je faisais une de ces suées ! je ne savais toujours pas que c'était la faute à Joe, vous pigez ? Je maudis les gars de la banque parce que je croyais que c'était de leur faute. Si Martine était simplement une blonde comme les autres... Mais elle a une sorte d'enthousiasme dévorant qui vous donne d'étranges sensations dans les genoux.

J'appelai donc les techniciens de la banque. l'écran dit : "Comment vous appelez-vous ?" Mais ça suffisait comme ça. Je tapai le nom d'un gars de la comptabilité et appris plein de choses très intéressantes. Je n'aurais jamais cru cela de lui. Puis le TI fit des boniments sur le nouveau service et me passa enfin la banque centrale. Je m'apprêtai à enguirlander le gars qui arriva sur l'écran, mais il me dit d'une voix lasse :



- Te voilà, Doucky ! Où es-tu en ce moment ? Quand peux-tu venir ? Tu ne peux pas savoir comme je pense souvent à toi !"

Je suis probablement le seul de ses copains qu'elle n'ait pas épousé un jour ou l'autre. Je fis de nouveau "Umph !" et j'avalai ma salive.

"Tu peux venir tout de suite ? demanda Martine joyeusement.

- C'est-à-dire que... euh... je travaille et... Ecoute, je... je te rappellerai.

"Laisse tomber. Tu as des ennuis mais tu n'es pas le seul. Qu'est-ce qu'ils t'ont fait encore ?"

Je le lui racontai, mais c'est tout juste s'il trouva ça drôle.

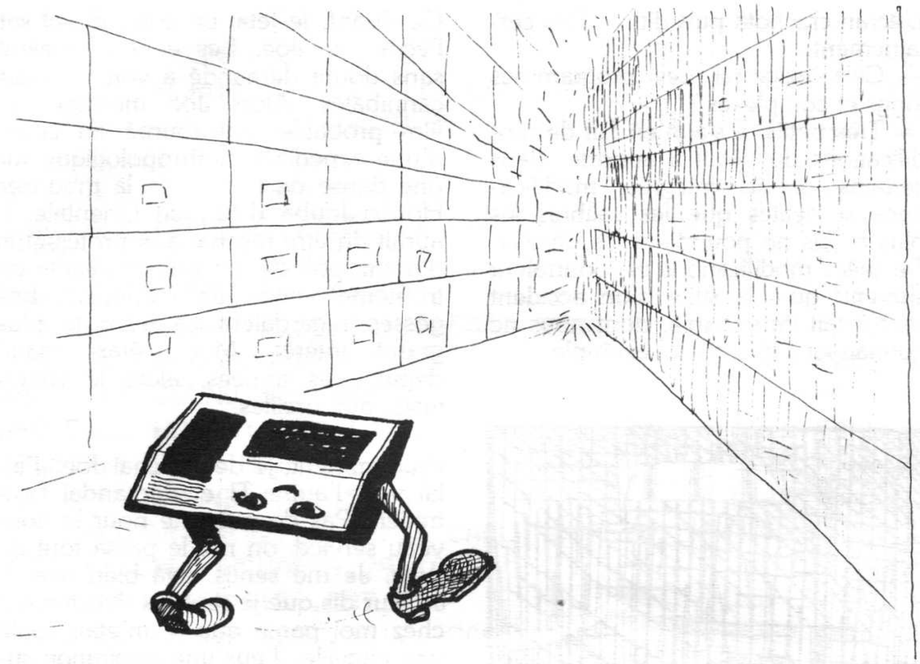
"Ce n'est rien, me dit-il, rien du tout. Nous venons juste de supprimer les fichiers contenant des renseignements sur les explosifs à grande puissance. Les demandes de conseils concernant la contrefaçon s'accroissent à chaque instant. Nous essayons aussi de détruire les renseignements ayant trait au meurtre parfait. Si les gens sont occupés à se renseigner les uns sur les autres, nous aurons peut-être le temps de bloquer les procédures de virements bancaires avant que tout le monde ne soit ruiné sauf les malins qui ont pensé à demander comment faire pour augmenter leur compte en banque.

- Mais faites donc quelque chose ! Pourquoi ne coupez-vous pas toutes les autres sources d'information ?

- Est-ce que tu te rends compte qu'il y a des années que l'industrie et les affaires ne fonctionnent que grâce aux banques de données ? Sans compter la distribution de télé, la météo, les horaires d'avion ou de train, la situation de l'emploi, toutes les communications... Les TI ont transformé la civilisation. Sans eux, nous sommes perdus ! Si tu crois que je ne deviens pas hystérique quand je pense à ce qui arrivera si ma femme apprend le chiffre réel de mon salaire et se met à se demander qui est cette rouquine..."

Il me regarda avec un sourire hagard et coupa la communication. Je m'assis et me pris la tête dans les mains. Il avait raison. Si les hommes des cavernes avaient dû renoncer à l'usage du feu, si ceux du XIXème siècle avaient dû renoncer à la vapeur et ceux du XXème siècle à l'électricité, que serait-il arrivé ? C'était pareil. Actuellement, un homme a besoin d'un ordinateur pour ses comptes et ses fichiers, d'un traitement de texte pour son courrier, du téléphone, de la télévision et des journaux, des bibliothèques, des encyclopédies, des dossiers, des annuaires, d'un avocat, d'un médecin, d'un dentiste, d'un pharmacien, d'une secrétaire... Pour tous les actes quotidiens de sa vie. Nous, nous n'avons besoin que d'un TI. Si nous voulons savoir quelque chose, parler à quelqu'un, nous appuyons sur les touches de notre TI. Si même nous voulons concevoir, programmer, créer des jeux ou autres applications, nous avons le TI. Mais Martine...

Je ne savais toujours pas ce qui s'était passé. Et maintenant, personne d'autre que moi ne le sait. Ce qui s'était passé, c'était Joe... L'ennui, c'était qu'il voulait faire son travail le mieux possible. Toutes ces histoires de conseils, de recettes, etc., nous aurions dû y penser bien avant. Trouver le moyen de mettre du poison dans la soupe de la femme d'un bonhomme, c'était la même chose, en un sens, que d'extraire une racine cubique ou de tenir un compte en banque. Dans tous les cas, il s'agissait de trouver la réponse à une question. Si tout allait mal, c'était parce que trop de réponses étaient données à trop de questions.



Un des TI du service s'alluma. J'y allai d'un pas las pour prendre la communication.
" Doucky ! " s'exclama Martine.
C'était la même chambre d'hôtel. Deux verres et une bouteille traînaient sur la table. L'un des verres était pour moi. Martine avait passé une robe de chambre à faire s'exorbiter un régiment d'aveugles. Elle me regardait avec ardeur.
" Doucky, je m'embête toute seule. Pourquoi ne viens-tu pas ?
— Je... (je m'étranglai.) Le travail...
— Pouah ! s'écria-t-elle. Doucky, tu ne te souviens pas des bons moments qu'on a passés ensemble ?...
J'eus un haut le corps.
" Tu es libre ce soir ? "
J'eus de nouveau un haut le corps parce qu'elle me sourit d'une façon qui tournerait certainement la tête à un célibataire, mais qui donne des frissons dans le dos à un homme respectablement marié comme moi.
" Doucky ! s'écria Martine (décidément elle raffolait de ce surnom ridicule !) J'ai été si méchante avec toi... Marions-nous ! "
Le désespoir me redonna de la voix.
" Je... suis déjà marié ", lui dis-je.
Elle ferma un instant les yeux puis, courageuse :
" Pauvre garçon ! On va te tirer de là, va. Ç'aurait été plus gentil si on avait pu se marier rapidement. Tant pis, on va se fiancer !
— Je... je ne peux pas...
— Je vais appeler ta femme, dit Martine, toute contente, et lui parler franchement. Si tu pouvais me donner le numéro de ton TI, chéri ? Comme ça, tout s'arrangerait... "

— *Clic !* Ça, c'était moi qui venais de couper la communication. Parfaitement, et j'ai même failli m'en évanouir. J'étais au bord de la dépression nerveuse.

Je me précipitai vers la porte en criant à quelqu'un qu'on m'avait appelé d'urgence. J'avais en projet de tourner en rond dans une des voitures de la boîte jusqu'à ce qu'il soit l'heure de rentrer à la maison. Ensuite, je prendrais femme et enfants et ficherais le camp quelque part où Martine ne me retrouverait jamais. Je n'avais aucune envie d'être son cinquième mari et peut-être sa seconde victime, un soir d'ennui. Je les connais, les blondes. Et je connaissais Martine ! Pour tout dire, j'avais une peur bleue.

Je pris une camionnette du service d'entretien et m'engageai dans le flot de la circulation. Derrière, il y avait un TI de rechange tout prêt pour le cas où il y aurait un échange standard à faire. Je conduisis comme un fou. La situation ne manquait pas d'ironie. Je perdais les pédales à cause d'un problème strictement personnel, pendant que la civilisation s'écroulait autour de moi parce que tous les gens voyaient leurs problèmes résolus sur-le-champ. On sait que les chercheurs de la Générale Électronique essayaient depuis longtemps de fabriquer un tube à filament non chauffé – autrement dit de résoudre le problème de l'émission à froid d'électrons. Un des chercheurs eut l'idée de soumettre cette question à un TI. Ce dernier fit la synthèse de quelques quintillions de faits connus et lui a donné la réponse. Pas plus difficile que de dire à une dame comment servir agréablement la soupe réchauffée ou comment se débarrasser sans fatigue du vieux meuble qui encombre le grenier.

Sans ce nouveau service des TI, Martine n'aurait jamais retrouvé ma trace. Mais maintenant, le feu était mis aux poudres... N'oubliez pas qu'elle a déjà tué un de ses maris et qu'elle a été acquittée ! Supposez qu'elle commence à s'impatienter et qu'elle demande au TI comment se débarrasser de ma femme de façon qu'on puisse se marier dans les quinze jours qui suivent ? Il le lui aurait dit ! De même qu'il a dit à une charmante banlieusarde comment s'assurer que son mari ne courrait plus jamais les jupons. Brrr ! Et le coup du gosse qui voulait savoir où trouver un trésor caché. Il était en train de déménager le contenu des coffres de Cartier lorsqu'ils l'ont attrapé. Le TI lui avait dit comment fabriquer une espèce de machine pour y accéder. On ne sait toujours pas comment ça fonctionne sinon qu'il doit y avoir une histoire de quatrième dimension là-dessous. Si Martine se mettait à poser des questions un peu plus techniques, les TI seraient à leur affaire ! Je peux vous dire que j'avais la trouille. Si vous croyez qu'il est indigne d'un homme d'avoir peur d'une seule blonde, c'est que vous n'avez jamais rencontré Martine.

Pendant que je conduisais comme un aveugle, un gars, conscient de ses responsabilités, demanda au TI comment appliquer dès maintenant son petit système personnel d'organisation sociale. Il ne lui demanda pas si son système était bon ou même s'il pouvait fonctionner. Non, il lui demanda simplement comment l'appliquer. Et le TI, ou Joe, le lui dit ! Au même moment, un prêtre en retraite demanda comment guérir la race du péché de concupiscence. Comme il avait soixante-dix berges, il était en dehors du coup, mais il voulait écarter ce péril mortel du restant de l'humanité. En trois secondes, la réponse arriva. Il suffisait de construire un émetteur diffusant des ondes modulées de façon particulière. C'est tout. Ça suffit. On a découvert le pot au rose par la suite, alors qu'il essayait de réunir les fonds pour passer à l'action. Heureusement, il n'avait pas eu l'idée de demander au TI comment financer l'opération, sinon nous aurions tous été guéris des élans que nous regrettons peut-être par la suite mais jamais sur le moment. Il y avait aussi le petit groupe de profonds penseurs qui était certain que la race humaine serait bien plus heureuse au fond des bois, en compagnie des fourmis et des champignons vénéneux. Ils ont demandé comment inciter les hommes à quitter les villes et les conditions de vie artificielles. Le nouveau service des TI a bien failli apporter une réponse définitive à la question !

Sur le moment, vous ne vous en êtes peut-être pas rendu compte, mais

pendant que je conduisais à l'aventure en suant sang et eau au sujet de Martine, le sort de la civilisation était en jeu. Je ne blague pas. Par exemple, tous les surhommes de la région demandaient au moyen de quelles armes ils pourraient prendre en main les affaires de l'humanité...

Moi, je conduisais sans savoir où j'allais, et je me dis :
" Ce que je devrais demander à ces cinglés de TI, c'est comment sortir de ce gâchis. Mais ce qu'ils me donneraient, c'est un moyen de me débarrasser sans risque de Martine. Je veux qu'on me fiche la paix! Je veux pouvoir devenir un menteur qui raconte aux jeunots quels trucs formidables il a faits, sans risquer de tout foutre par terre en les faisant vraiment. "

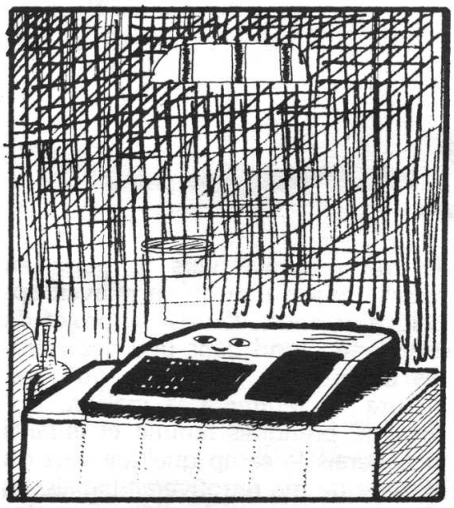
Je tournai à droite, au hasard dans ma camionnette.
" Oui, me dis-je, amer. Le monde était beau jadis. Je pouvais rentrer chez moi sans avoir des crampes d'estomac à force de me demander si une blonde ne vient pas d'appeler ma femme pour lui annoncer nos fiançailles. Je pouvais me servir d'un TI sans voir apparaître une chambre d'hôtel dans laquelle la même blonde s'aère l'épiderme, tournant mes pensées vers des choses que je dois éviter.
Peut-être... "

Puis je me souvins que ma femme allait certainement dire que c'était ma faute si quelqu'un apprenait un jour tout sur notre vie privée. " C'était chouette, ai-je pensé, en me rappelant les beaux jours d'avant-hier matin. On était heureux, et voilà qu'une sacrée machine est arrivée à tout fracasser. "

C'est à ce moment que je compris tout en un éclair. La banque n'est pas construite de manière à pouvoir manipuler elle-même ses propres programmes. Les programmes sont exclusivement manipulés par des TI et protégés par des clés particulières. Seul un TI peut y accéder pour fabriquer ce nouveau service. Les hommes n'y seraient jamais arrivés. Seul un TI peut avoir assimilé toutes les données nécessaires pour faire travailler les autres TI de cette façon...

Il n'y avait qu'un moyen de s'en assurer. Je m'arrêtai devant un restaurant, me dirigeai vers le TI et mis une pièce dans la fente. Puis je tapai :
" Est-ce qu'un TI peut être programmé de façon à coopérer dans des prévisions à longue échéance trop compliquées pour le cerveau humain ? "

L'écran crachota puis dit : " Très certainement.
— Cela exige-t-il des programmes longs et complexes ?
— Très petits. Il s'agit en fait de modifications de détail à apporter dans certains circuits intégrés, modifications si petites que les calibres les plus précis ne pourraient les déceler. De telles modifications ne pourraient survenir qu'à la suite d'un accident tout à fait improbable, dont nous ne connaissons qu'un seul exemple.



- Ce TI accidentellement modifié peut accomplir des tâches d'une très grande importance. Est-il possible de le retrouver ?"

L'écran crachota. J'étais trempé de sueur. J'avais peur que mon Joe ne devienne méfiant. Mais la question que j'avais posée était parfaitement logique, et les TI ne savent pas mentir. Ils disent toujours la vérité; ils n'y peuvent rien.

"Un TI complet capable de faire le travail demandé se trouve actuellement dans la famille..."

Et il me donna l'adresse. J'y courus comme un dératé. Je rangeai ma camionnette devant l'immeuble, un H.L.M. de proche banlieue, et montai les escaliers à toute allure. Je sonnai et un gosse vint m'ouvrir.

"Je suis de service d'entretien, lui dis-je. Nos fiches indiquent que votre TI peut avoir une panne d'un moment à l'autre. Alors je suis venu le remplacer."

"D'accord !" répondit le gosse, tout heureux. Et il retourna au living où ses frères et soeurs étaient en train de regarder une émission. Je branchai l'autre TI et, consciencieux, m'assurai qu'il fonctionnait bien. Puis je leur dis :

"Voilà, les gosses. Maintenant vous vous servez de celui-là et j'enlève le vieux."

Ce disant, je jetai un coup d'oeil sur l'écran de Joe. Les enfants avaient sans doute demandé à voir de vrais cannibales. Alors Joe montrait un film probablement tourné au cours d'une expédition anthropologique sur une danse de fertilité de la tribu des Houba-Jouba d'Afrique Orientale. Il aurait dû être réservé aux professeurs d'anthropologie ou aux étudiants en troisième année de médecine. Les gosses regardaient ça avec le plus grand intérêt. Moi, j'étais marié depuis des années, alors je rougis jusqu'aux oreilles.

Prudemment, je débranchai Joe. J'allai vers l'autre TI et demandai l'entretien. Pas de publicité pour le nouveau service; on me le passa tout de suite. Je me sentis déjà bien mieux. Je leur dis que je rentrais directement chez moi parce que je m'étais foulé une cheville. J'eus une inspiration subite et ajoutai :

"Dites donc, en tombant j'ai cassé le TI que je venais de remplacer. Je l'ai laissé sur place pour que les boueux le ramassent.

- Si tu ne le ramènes pas pour qu'on en récupère les pièces, tu devras le payer, me dirent-ils.

- Ca vaut le prix !"

Je rentrai chez moi. Martine n'avait pas appelé. J'allai tout de suite porter Joe à la cave. Si je le rapportais à l'atelier, ils récupèreraient les pièces et celle qui n'est pas dans la norme serait utilisée de nouveau pour une réparation. C'était un trop gros risque. Je l'ai payé et, ainsi, le problème a été réglé.

Voilà. Si vous dites que j'ai sauvé la civilisation humaine, vous ne risquez pas de vous tromper. Tant que Martine vivra, je peux pas prendre le risque de le remettre en fonctionnement. Et puis ce n'est pas tout. Il y a les timbrés qui veulent transformer le monde, et tous ceux qui ont des "problèmes" à résoudre. C'est embêtant les problèmes, bien sûr, mais il vaut peut-être mieux les laisser comme ils sont.

D'un autre côté, on pourrait peut-être dompter Joe pour le faire travailler raisonnablement. Il n'aurait pas de mal à me procurer un ou deux milliards de francs, même légers. Mais même si j'ai la sagesse de ne pas désirer la richesse et que je prends paisiblement ma retraite... Enfin ! Ca me plaira peut-être, ou peut-être pas. Si jamais j'en ai assez d'être vieux et de ne plus être capable d'autre chose que de penser, je pourrai peut-être rebrancher Joe,

juste le temps de lui demander : "Comment faire pour ne plus être vieux ?" Il trouverait sûrement le moyen de me donner la réponse.

Pour le moment, depuis la fin de cette histoire, les TI se comportent de nouveau convenablement. C'est fou le nombre de services qu'ils peuvent rendre si on les utilise bien, même sans parler du "service spécial" que Joe avait mis en place. Chaque maison, chaque appartement possède son TI. On en trouve même dans les rues, à la place où, il y a seulement cinq ans, il y avait encore de ces vieilles cabines téléphoniques dont les appareils étaient toujours cassés.

Plus aucune famille ne peut se passer de sa machine, que ce soit pour communiquer au moyen du service de vidéophonie, ou pour mémoriser toutes sortes d'informations : recettes de cuisine, courrier, factures, compte de chèques etc. Des gosses jouent à la bataille navale d'une ville à l'autre. La banque centrale donne chaque jour des idées de menu en fonction de la saison et du budget de la ménagère. Et je ne parle pas des renseignements et réservations sur les avions et les trains, service qui fut l'un des premiers installé sur les Terminaux Intelligents. Et si quelques maniaques veulent se défouler, ils peuvent toujours se programmer leurs propres applications, si toutefois celles qui leur sont proposées par la banque ne leur conviennent pas.

A y repenser, j'ai des frissons dans le dos quand je me souviens que ce sacré Joe a failli mettre tout ça en l'air...



TI-Writer à l'essai

Thomas Coppens

Vous avez sans doute longtemps traité vos textes comme moi, c'est-à-dire tout simplement avec une paire de ciseaux, un flacon de gomme arabique, celui-ci remplacé plus tard par le bâton de colle. Le résultat de ces découpages et collages était une feuille de papier à plusieurs épaisseurs ressemblant plus à une carte en relief du Massif Central qu'à un texte devant partir chez l'imprimeur. En outre, les ratures et corrections, plutôt abondantes, rendaient le texte parfois difficile à lire. Cela n'était cependant pas très grave puisque la première épreuve de l'imprimeur devait malgré tout être entièrement relue et comptait souvent plus de fautes que l'original. Ne vous est-il jamais arrivé de trouver dans la cinquième épreuve des fautes que vous n'aviez jamais remarquées dans les épreuves précédentes ?

Tout cela est désormais du passé : la machine à traitement de textes résout à présent ce genre de problèmes. Il subsistait cependant la question du prix de cette machine : un équipement professionnel dans ce domaine coûtait environ 30.000 francs (ou plus) sans compter le programme de traitement de texte lui-même. En outre, on n'avait pas pour ce prix de programme comportant l'accentuation propre à la langue française.

Les possesseurs du TI-99/4A possèdent déjà une partie de l'équipement. Il leur manque peut-être quelque chose, mais ils ont le matériel de base pour utiliser le programme TI-WRITER. Il s'agit là d'un excellent programme de traitement de texte coûtant environ 1.000 francs aux Etats-Unis. La configuration nécessaire pour l'utiliser est la suivante :

- console TI-99/4A;
- boîtier d'expansion;
- carte d'extension de mémoire;
- un lecteur de disque avec son contrôleur;
- interface RS232;
- imprimante à 80 caractères par ligne.

Vous objecterez sans doute qu'une machine de traitement de texte avec un écran ne permettant de placer que 28 caractères par ligne, cela ne peut guère être intéressant. Erreur : le TI-WRITER peut aussi travailler avec 40 caractères par ligne. Vous ne disposerez évidemment pas des lutins (sprites), mais vous n'en avez pas vraiment besoin lorsque vous traitez un texte. Savez-vous à ce sujet qu'il existe un ensemble de programmes permettant de disposer en BASIC de 40 caractères par ligne? Le VDP (Video Display Processor) tient en fait bien d'autres propriétés en réserve et l'on peut affirmer sans crainte de se tromper que les possibilités du TI-99/4A sont loin d'être épuisées. Il est d'ailleurs dommage que la plupart des possesseurs de cet appareil ne s'en servent que pour des jeux électroniques.

Mais revenons à ce qui nous occupe : comment se présente le TI-WRITER? Il s'agit cette fois encore d'un module à enficher dans la console, tout comme un simple module de jeu ou comme l'Extended Basic. Dès l'allumage de la machine, module en place, apparait à l'écran l'image d'essai puis le choix entre TI-BASIC ou l'une des sept langues de TI-WRITER. Le choix d'une langue détermine l'utilisation d'une police de caractères (une police de caractères est l'ensemble des caractères disponi-

bles sur une machine à un moment donné) propre à cette langue. En ce qui concerne le français, vous disposerez donc des accents ainsi que du "ç" sur l'écran. Formidable, non?

Pour commencer l'exploration de ce programme, commençons à taper un texte et constatons de suite notre première surprise : on dispose de fenêtres à l'écran. Celui-ci permet en effet non seulement de lire 40 caractères par ligne mais aussi les colonnes 20 à 60 ou 40 à 80. Le texte peut par conséquent comporter 80 caractères par ligne.

Le TI-WRITER est accompagné d'un manuel complet ainsi que d'une disquette de démonstration. La version dont je dispose est d'origine américaine et donc rédigée en anglais. Texas Instruments France s'occupe actuellement de faire traduire le manuel et la disquette : un peu de patience donc!

Il est recommandé de lire très attentivement le manuel avant de commencer à taper son premier texte. Au fur et à mesure de cette lecture, vous serez étonné de toutes les possibilités qui vous sont offertes. Vous pourrez par exemple changer la couleur des caractères et du fond sur lequel ils apparaissent, rechercher automatiquement un mot dans le texte (il faut moins d'une seconde pour trouver le mot "XYZ" disposé en fin d'un texte long en partant de la première ligne : peut-on encore parler de la lenteur du TI-99?). Vous pouvez aussi remplacer un groupe de caractères par un autre et pour cela vous disposez de trois méthodes possibles :

- vous remplacez seulement le premier groupe de caractères rencontré correspondant à celui que vous recherchez;
- vous remplacez tous les groupes de caractères correspondant et se trouvant dans le texte;
- vous décidez au coup par coup; c'est-à-dire que l'ordinateur recherche chaque groupe correspondant et vous demande à chaque fois s'il doit procéder à la substitution.

Vous pourrez aussi, bien entendu, supprimer une partie du texte ou insérer du texte supplémentaire, supprimer ou insérer un ou plusieurs paragraphes, recopier automatiquement une ligne, placer des tabulations, vous pouvez, vous pouvez... Qu'est-ce donc qui n'est pas possible?

Combien de caractères pensez-vous pouvoir traiter? Ne tombez pas à la renverse en apprenant que la réponse est de 23.000 caractères (il faut donc disposer de l'extension mémoire de 32K pour exploiter toutes les possibilités du TI-WRITER.).



Que peut-il faire de plus ?

- déplacer des paragraphes entiers ;
- insérer des titres ou sous-titres dans le texte ;
- rédiger des lettres personnalisées combinées avec une liste d'adresses ;
- avec la commande "RECOVER EDIT", retrouver un texte effacé par mégarde (il vous faudra tout de même un minimum d'expérience !).

Lorsque votre texte est tapé sur l'écran, vous avez la possibilité de le faire imprimer sur papier sous une forme plus élaborée. Vous disposez pour cela de plusieurs fonctions :

- numérotation automatique des pages ;
- détermination du nombre de lignes à mettre dans chaque page ;
- insertion d'un en-tête après chaque saut de page ;
- insertion d'une note en bas d'une page comportant un renvoi ;
- centrage d'une partie de texte ;
- impression en caractères gras ;
- marge à droite par insertion automatique de "blancs" supplémentaires...

On dispose en tout de 28 commandes de mise en page !

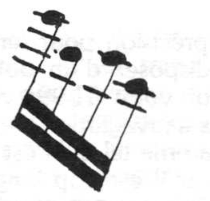
La division automatique des mots est impossible, le TI-WRITER ne comportant pas les règles de césure propres à chaque langue. Il ne divise donc pas les mots et reporte un mot trop long au début de la ligne suivante (cette possibilité peut cependant être mise hors fonction).

Je ne peux conclure qu'en affirmant encore une fois l'excellente qualité du programme TI-WRITER qui exploite à fond les possibilités du processeur 9900. Le programme est d'ailleurs écrit par Microsoft, une société mondialement connue pour ses logiciels micro-informatiques : une bonne garantie de qualité. Il est d'un maniement très facile une fois qu'on l'a bien en main. Si vous doutez des possibilités du TI-99, offrez-vous le TI-WRITER et vous changerez d'avis!

Il est intéressant de savoir aussi que le TI-WRITER peut utiliser les tableaux de chiffres rédigés au moyen du programme Multiplan, autre produit de Microsoft disponible sur le TI-99. Enfin, il faut bien souligner que le texte se présente à l'écran de la même façon qu'il sera imprimé sur le papier, ce qui n'est pas le cas de tous les programmes de traitement de texte tournant sur micro-ordinateurs.

NDLR : *L'enthousiasme de notre ami Thomas Coppens (notre rédacteur en Belgique) ne peut être conforté, ou compensé, par nous car nous n'avons pas encore eu l'occasion de tester le TI-WRITER. Dès que nous le pourrons, nous apporterons notre grain de sel à cette analyse bien sympathique.*





Canon de Purcell

Claire Arribet

Henry Purcell était un musicien anglais du 17ème siècle, sans doute le plus grand musicien que l'Angleterre ait enfanté. Le programme que je vous présente n'est donc pas un jeu guerrier, comme son titre pourrait le laisser croire, mais un programme faisant jouer de la musique par le TI-99.

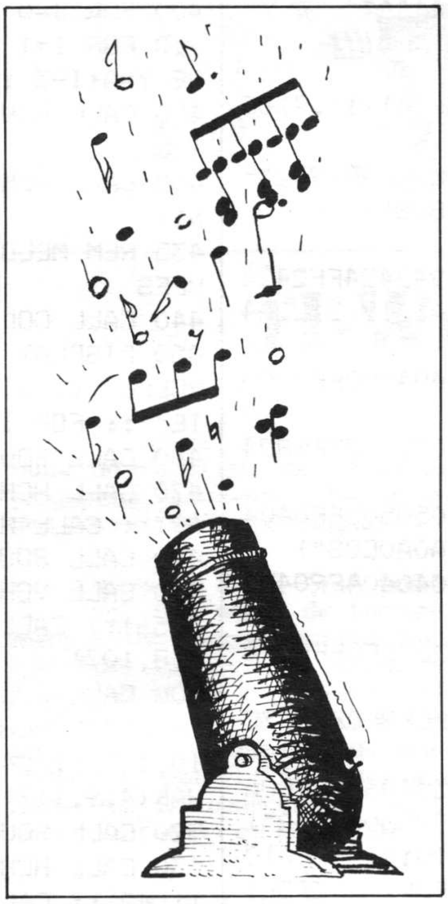
Je ne savais trop comment occuper mes dernières vacances de Pâques. Alors, après quelques disputes avec les membres de ma famille, j'ai monopolisé pendant cinq jours entiers notre ordinateur en me fixant pour but de lui faire jouer cette musique tout en présentant la partition à l'écran. J'ai commencé par travailler sur la partition elle-même. En premier lieu, j'ai recensé tous les caractères dont je devais disposer pour construire le graphisme. Il y a bien entendu les lignes des portées : il me fallait 4 lignes de portée, c'est-à-dire quatre séries de 5 lignes horizontales. Cela suppose bien entendu quatre clés de sol.

Il me faut ensuite des notes. Sur la partition, cela ne semble pas poser de problème particulier. Il y en a cependant un de taille : lorsque l'on dispose une note sur une ligne, on efface obligatoirement un bout de cette ligne de chaque côté de la note. Il faut donc dessiner plusieurs catégories de notes :

- note à disposer entre deux lignes ;
- note à disposer sur une ligne ;
- note à disposer au dessus de la portée ;
- note à disposer en-dessous de la portée.

Une note est composée d'un rond noir surmonté d'une queue différente selon qu'il s'agit d'une "noire" ou d'une "croche". Il me faut donc fabriquer quatre types de queues : noire vers le haut, noire vers le bas, puis croche vers le haut ou vers le bas. Enfin, dernier caractère à concevoir, le dièse : pas de difficulté car c'est le caractère 35 dans le code ASCII du TI-99. Tous ces caractères sont construits dans les lignes 200 à 340 du programme.

Ceci étant fait, il est alors possible de faire jouer une première fois la mélodie. Je commence par tracer la portée sur l'écran (lignes 350 à 390) et y apposer les clés de sol (lignes 400 à 430). Puis je fais jouer cette mélodie, sur une seule voie sonore, en faisant apparaître les notes sur les lignes de la portée au fur et à mesure qu'elles sont exécutées. L'exécution de chaque note comporte donc une instruction CALL SOUND suivie d'autant de CALL VCHAR ou CALL HCHAR qu'il est nécessaire pour dessiner le signe musical correspondant. Le CALL SOUND est en



premier car il ne bloque pas le déroulement du programme : le TI-99 "lance" l'exécution de la note et, alors même que cette exécution se poursuit, il passe à l'instruction suivante, c'est-à-dire qu'il effectue de manière simultanée les CALL VCHAR ou HCHAR.

Toute la mélodie se joue ainsi (lignes 440 à 1530) puis, après un délai d'attente (ligne 1530), débute le canon proprement dit. Pour ceux qui ne le sauraient pas, je rappelle ce qu'en termes musicaux représente un canon : plusieurs chanteurs ou instruments sont présents. L'un d'eux commence l'exécution d'un morceau. Lorsqu'il a terminé la première phrase de ce morceau et entame la seconde, le deuxième musicien commence à son tour l'exécution de la première. Le troisième décale ensuite son départ du temps d'une phrase musicale, et ainsi de suite (en fait il est rare de voir des canons à plus de trois voix.).

Le programme commence par modifier la couleur du fond de l'écran et affiche le mot "CANON" au bas de celui-ci (ligne 1540). J'ai choisi de ne pas effacer les notes des lignes de portée, mais d'indiquer au moyen d'un lutin de couleur (sprite) la note sur laquelle se trouve chacune des trois voies sonores. J'ai donc défini trois lutins en forme de tête de flèche orientée vers le haut pour remplir cette fonction (lignes 1550 et 1560).

Le jeu du canon commence ensuite par la première voie sonore. Chacune des lignes d'instructions est composée d'un CALL SOUND suivi d'un CALL LOCATE qui détermine la place du lutin correspondant, juste sous la note qui est jouée (lignes 1570 à 1730). Puis la deuxième voie sonore entre en jeu : les instructions deviennent un peu plus complexes (lignes 1740 à 1900).

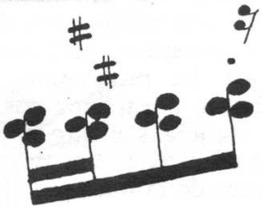
La suite du morceau peut être programmée différemment. En effet, chaque voie reprend ce que vient de jouer la voie précédente. Il est alors aisé de composer une boucle dans laquelle une voie joue toujours la même mélodie mais où seule la couleur des lutins change. De la sorte apparaissent bien à l'écran trois lutins de couleurs différentes se déplaçant alors que l'on entend trois voies sonores (lignes 1910 à 2170).

La ligne 2180 exécute l'accord final, puis, après un temps d'attente (ligne 2190), on efface les lutins et on renvoie le programme à son début (ligne 2200). On ne fait évidemment pas GOTO 100 mais GOTO 360 car il est inutile de préciser une deuxième fois la forme des caractères qui restent définis tant que l'on n'a pas arrêté le programme.

Je ne voudrais surtout pas avoir l'air de me vanter, mais je constate que tous ceux qui ont vu tourner ce programme l'ont trouvé très spectaculaire. Je crois que le mode d'écriture que j'ai choisi est perfectible. Il existe certainement une autre façon de concevoir un programme pour l'exécution de ce canon. Celle que j'ai choisie m'a semblé présenter l'avantage de la simplicité : la lecture seule du listing donne une bonne idée de ce que fait le programme.

Une dernière précision pour terminer : si vous disposez d'un boîtier d'expansion sur votre TI-99, vous ne pourrez pas sauvegarder et rappeler ce programme tel qu'il est sur une cassette, car il est trop long. Il faut donc en taper une première moitié, la sauvegarder, puis taper la deuxième moitié et la sauvegarder avec MERGE. Pour rappeler ensuite le programme à partir du magnétophone, on entre la première moitié normalement et la deuxième avec MERGE.

NDLR : *sur la cassette d'accompagnement de ce numéro (cassette 99-2), le programme est enregistré en entier sans MERGE. Les possesseurs d'un boîtier d'expansion devront débrancher celui-ci avant de commencer le chargement du programme.*





```
100 REM ********************
110 REM * CANON DE PURCELL *
120 REM ********************
130 REM *************************
140 REM * COPYRIGHT 99 MAGAZINE *
150 REM *  ET CLAIRE ARRIBET     *
160 REM *************************
170 REM
180 CALL CLEAR
190 CALL SCREEN(10):: DISPLAY AT(11,7)BE
EP:"CANON DE PURCELL"
200 CALL CHAR(96,"FF"):: CALL CHAR(97,"F
F01010101010101FF0808080808080808")
205 REM CLE DE SOL(40 A 47)
210 CALL CHAR(40,"FF18282424242424FF2424
242424242428FF302060A0203C62FFA1A121212 2A4
78")
220 CALL CHAR(44,"FF202020A0A04000FF0000
000010102FF0202020201")
230 CALL CHAR(60,"FF1824241C042418FF1824
243C42423C")
240 CALL CHAR(48,"FF00000406050505FF0404
0404040404FF8080808080808080FFA0A0C08")
250 CALL CHAR(52,"FF0605050504040404FF0404
04043C7CFCFFF87")
260 CALL CHAR(56,"FF000000000387CFCFFF8F0
8080808080FF8080A0A0A0C08")
270 CALL CHAR(88,"FF00387CFCFCF8F0FF3C7C
FCFCF87")
280 CALL CHAR(104,"FF0004060504060 5FF80A
0C080A0C080FF00000404040404FF808080808")
290 CALL CHAR(112,"FF00000101010101FF010
101010101 01FF0000008040404 0FF0F1F3F3F3E1
C")
300 CALL CHAR(116,"FF000002030282828FF010
30707070300FFE0E0E0E0C08")
310 CALL CHAR(120,"FF0080808000000 0FF030
2020202020 2FF0202FF021E3E7EFF7C38")
320 CALL CHAR(124,"FF000020202020 20FF202
0202020202 0FF060504060504040000000000387
CFC")
330 CALL CHAR(128,"FF090909090909 09FF090
96969090909FF808080808080808")
340 CALL CHAR(131,"FF161C34161C341400000
01818081000FF00000000C0C000FF00C0C")
350 FOR X=1 TO 500 :: NEXT X
355 REM TRACE DE LA PORTEE
360 CALL CLEAR :: CALL SCREEN(11)
370 FOR I=2 TO 25 STEP 6
380 CALL HCHAR(I,1,96,160):: CALL VCHAR(
I,32,97,4)
390 NEXT I
395 REM TRACE DES CLES DE SOL
400 FOR J=0 TO 3 :: A=1+6*J :: X=3
410 FOR I=1 TO 7 :: IF I<6 THEN Y=A+I EL
SE Y=A+I-3 :: X=2
420 CALL HCHAR(Y,X,39+I):: NEXT I :: NEX
T J
430 CALL HCHAR(3,4,60):: CALL HCHAR(4,4,
61)
435 REM MELODIE ET ECRITURE DES NOTES JO
UEES
440 CALL COLOR(5,14,11,6,14,11,7,14,11)
450 DISPLAY AT(1,1)SIZE(16):"CANON DE PU
RCELL" :: DISPLAY AT(24,4)SIZE(7):"MELOD
IE" :: FOR I=1 TO 150 :: NEXT I
460 CALL SOUND(300,440,0)
470 CALL HCHAR(2,6,48):: CALL HCHAR(3,6,
49):: CALL HCHAR(4,6,89)
480 CALL SOUND(600,587,0)
490 CALL VCHAR(2,7,98,4):: CALL HCHAR(2,
8,56):: CALL HCHAR(3,8,57):: CALL HCHAR(
4,8,107)
500 CALL SOUND(300,440,0)
510 CALL HCHAR(2,10,116):: CALL HCHAR(3,
10,125):: CALL HCHAR(4,10,118):: CALL HC
HAR(4,9,117)
520 CALL SOUND(600,349,0)
530 CALL HCHAR(3,11,106):: CALL HCHAR(4,
11,49):: CALL HCHAR(5,11,89)
540 CALL SOUND(300,294,0)
550 CALL HCHAR(4,13,48):: CALL HCHAR(5,1
3,49):: CALL HCHAR(6,13,89)
560 CALL SOUND(600,440,0)
570 CALL HCHAR(2,15,106):: CALL HCHAR(3,
15,49):: CALL HCHAR(4,15,89)
580 CALL SOUND(300,494,0)
590 CALL HCHAR(2,17,52):: CALL HCHAR(3,1
7,53):: CALL HCHAR(4,17,54):: CALL VCHAR
(2,18,98,4)
600 CALL SOUND(600,523,0)
610 CALL HCHAR(3,19,88):: CALL HCHAR(4,1
9,50):: CALL HCHAR(5,19,107):: CALL HCHA
```

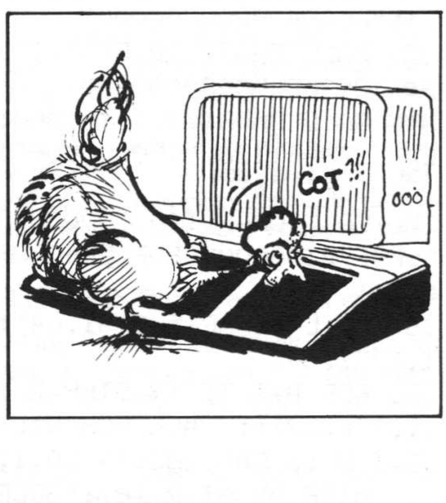
```
R(1,20,132)
620 CALL SOUND(300,554,0)
630 CALL HCHAR(3,20,131):: CALL HCHAR(3,
21,88):: CALL HCHAR(4,21,50):: CALL HCHA
R(5,21,51)
640 CALL SOUND(600,587,0)
650 CALL HCHAR(2,23,56):: CALL HCHAR(3,2
3,57):: CALL HCHAR(4,23,107)
660 CALL SOUND(300,440,0)
670 CALL HCHAR(2,25,48):: CALL HCHAR(3,2
5,49):: CALL HCHAR(4,25,89)
680 CALL SOUND(600,698,0)
690 CALL HCHAR(1,27,127):: CALL HCHAR(2,
27,57):: CALL HCHAR(3,27,107)
700 CALL SOUND(150,659,0)
710 CALL HCHAR(2,29,88):: CALL HCHAR(3,2
9,50):: CALL HCHAR(4,29,105)
720 CALL SOUND(150,587,0)
730 CALL HCHAR(2,31,56):: CALL HCHAR(3,3
1,57):: CALL HCHAR(4,31,105)
740 CALL SOUND(600,587,0)
750 CALL HCHAR(8,5,56):: CALL HCHAR(9,5,
57):: CALL HCHAR(10,5,107)
760 CALL SOUND(300,554,0)
770 CALL HCHAR(9,7,131):: CALL HCHAR(9,8
,88):: CALL HCHAR(10,8,50):: CALL HCHAR(
11,8,51)
780 CALL SOUND(600,587,0)
790 CALL HCHAR(8,10,56):: CALL HCHAR(9,1
0,57):: CALL HCHAR(10,10,107):: CALL HCH
AR(7,11,132)
800 CALL SOUND(300,554,0)
810 CALL HCHAR(9,12,131):: CALL HCHAR(9,
13,88):: CALL HCHAR(10,13,50):: CALL HCH
AR(11,13,51)
820 CALL SOUND(600,587,0)
830 CALL HCHAR(8,15,56):: CALL HCHAR(9,1
5,57):: CALL HCHAR(10,15,107)
840 CALL SOUND(300,659,0)
850 CALL HCHAR(8,17,88):: CALL HCHAR(9,1
7,50):: CALL HCHAR(10,17,51)
860 CALL SOUND(600,698,0)
870 CALL VCHAR(8,18,97,4):: CALL HCHAR(7
,20,127):: CALL HCHAR(8,20,57):: CALL HC
HAR(9,20,107)
880 CALL SOUND(300,392,0)
890 CALL HCHAR(9,22,52):: CALL HCHAR(10,
22,53):: CALL HCHAR(11,22,54)
900 CALL SOUND(600,349,0)
910 CALL HCHAR(9,24,106):: CALL HCHAR(10
,24,49):: CALL HCHAR(11,24,89)
920 CALL SOUND(300,294,0)
930 CALL HCHAR(10,27,114):: CALL HCHAR(1
0,26,112):: CALL HCHAR(11,26,113):: CALL
 HCHAR(12,26,115)
940 CALL SOUND(600,440,0)
950 CALL HCHAR(8,29,124):: CALL HCHAR(9,
29,125):: CALL HCHAR(10,29,118):: CALL H
CHAR(10,28,117):: CALL HCHAR(7,30,132)
960 CALL SOUND(300,392,0)
970 CALL HCHAR(9,31,52):: CALL HCHAR(10,
31,53):: CALL HCHAR(11,31,54)
980 CALL SOUND(600,349,0)
990 CALL HCHAR(15,5,106):: CALL HCHAR(16
,5,49):: CALL HCHAR(17,5,89)
1000 CALL SOUND(300,330,0)
1010 CALL HCHAR(16,7,52):: CALL HCHAR(17
,7,53):: CALL HCHAR(18,7,54)
1020 CALL SOUND(600,294,0)
1030 CALL HCHAR(16,9,106):: CALL HCHAR(1
7,9,49):: CALL HCHAR(18,9,89)
1040 CALL SOUND(300,349,0)
1050 CALL HCHAR(15,11,48):: CALL HCHAR(1
6,11,49):: CALL HCHAR(17,11,89)
1060 CALL SOUND(600,440,0)
1070 CALL HCHAR(14,13,106):: CALL HCHAR(
15,13,49):: CALL HCHAR(16,13,89)
1080 CALL SOUND(300,220,0)
1090 CALL HCHAR(17,16,120):: CALL HCHAR(
17,15,121):: CALL HCHAR(18,15,122):: CAL
L HCHAR(19,15,123)
1100 CALL SOUND(600,294,0)
1110 CALL VCHAR(14,17,130,4):: CALL HCHA
R(16,18,124):: CALL HCHAR(17,18,125):: C
ALL HCHAR(18,18,118):: CALL HCHAR(18,17,
117)
1120 CALL SOUND(300,659,0)
1130 CALL HCHAR(13,19,132):: CALL HCHAR(
14,20,88):: CALL HCHAR(15,20,50):: CALL
HCHAR(16,20,51)
1140 CALL SOUND(600,698,0)
1150 CALL HCHAR(13,22,127):: CALL HCHAR(
14,22,57):: CALL HCHAR(15,22,107)
1160 CALL SOUND(300,554,0)
1170 CALL HCHAR(15,24,131):: CALL HCHAR(
15,25,88):: CALL HCHAR(16,25,50):: CALL
HCHAR(17,25,51)
1180 CALL SOUND(450,587,0)
1190 CALL HCHAR(14,27,56):: CALL HCHAR(1
5,27,57):: CALL HCHAR(16,27,58):: CALL H
CHAR(14,28,133)
1200 CALL SOUND(150,659,0)
1210 CALL HCHAR(14,29,88):: CALL HCHAR(1
5,29,50):: CALL HCHAR(16,29,105)
1220 CALL SOUND(300,698,0)
1230 CALL HCHAR(13,31,127):: CALL HCHAR(
14,31,57):: CALL HCHAR(15,31,58)
1240 CALL SOUND(450,587,0)
1250 CALL HCHAR(20,5,56):: CALL HCHAR(21
,5,57):: CALL HCHAR(22,5,58):: CALL HCHA
R(20,6,133)
1260 CALL SOUND(150,554,0)
1270 CALL HCHAR(21,7,131):: CALL HCHAR(2
1,8,88):: CALL HCHAR(22,8,50):: CALL HCH
AR(23,8,105)
1280 CALL SOUND(300,587,0)
1290 CALL HCHAR(20,10,56):: CALL HCHAR(2
1,10,57):: CALL HCHAR(22,10,58)
```

```
1300 CALL SOUND(600,659,0)
1310 CALL HCHAR(20,12,88):: CALL HCHAR(2
1,12,50):: CALL HCHAR(22,12,107):: CALL
HCHAR(19,13,132)
1320 CALL SOUND(300,659,0)
1330 CALL HCHAR(20,14,88):: CALL HCHAR(2
1,14,50):: CALL HCHAR(22,14,51)
1340 CALL SOUND(600,440,0)
1350 CALL HCHAR(20,16,124):: CALL HCHAR(
21,16,125):: CALL HCHAR(22,16,118):: CAL
L HCHAR(22,15,117)
1360 CALL SOUND(300,392,0)
1370 CALL HCHAR(21,17,52):: CALL HCHAR(2
2,17,53):: CALL HCHAR(23,17,54):: CALL V
CHAR(20,18,98,4)
1380 CALL SOUND(450,349,0)
1390 CALL HCHAR(21,19,48):: CALL HCHAR(2
2,19,49):: CALL HCHAR(23,19,89):: CALL H
CHAR(23,20,134)
1400 CALL SOUND(150,392,0)
1410 CALL HCHAR(21,21,126):: CALL HCHAR(
22,21,53):: CALL HCHAR(23,21,54)
1420 CALL SOUND(300,440,0)
1430 CALL HCHAR(20,23,48):: CALL HCHAR(2
1,23,49):: CALL HCHAR(22,23,89)
1440 CALL SOUND(450,349,0)
1450 CALL HCHAR(21,25,48):: CALL HCHAR(2
2,25,49):: CALL HCHAR(23,25,89):: CALL H
CHAR(23,26,134)
1460 CALL SOUND(150,330,0)
1470 CALL HCHAR(22,27,126):: CALL HCHAR(
23,27,53):: CALL HCHAR(24,27,54)
1480 CALL SOUND(300,294,0)
1490 CALL HCHAR(22,29,48):: CALL HCHAR(2
3,29,49):: CALL HCHAR(24,29,89)
1500 CALL SOUND(2000,587,0,294,0,147,5)
1510 CALL HCHAR(22,31,106):: CALL HCHAR(
23,31,49):: CALL HCHAR(24,31,89)
1520 CALL VCHAR(20,32,128,4)
1530 FOR F=1 TO 1000 :: NEXT F
1535 REM CANON ET INDICATION DES NOTES J
OUEES(LUTINS)
1540 CALL SCREEN(12):: CALL COLOR(5,14,1
2,6,14,12,7,14,12):: DISPLAY AT(24,4)SIZ
E(7):" CANON" :: CALL VCHAR(21,32,129,2)
1550 CALL CHAR(94,"0010387CEE")
1560 CALL SPRITE(#1,94,9,200,30):: CALL
SPRITE(#2,94,3,200,40):: CALL SPRITE(#3,
94,5,200,50)
1570 CALL SOUND(300,440,0):: CALL LOCATE
(#1,32,41)
1580 CALL SOUND(600,587,0):: CALL LOCATE
(#1,32,57)
1590 CALL SOUND(300,440,0):: CALL LOCATE
(#1,32,70)
1600 CALL SOUND(600,349,0):: CALL LOCATE
(#1,40,81)
1610 CALL SOUND(300,294,0):: CALL LOCATE
(#1,48,97)
1620 CALL SOUND(600,440,0):: CALL LOCATE
(#1,32,113)
1630 CALL SOUND(300,494,0):: CALL LOCATE
(#1,28,129)
1640 CALL SOUND(600,523,0):: CALL LOCATE
(#1,36,145)
1650 CALL SOUND(300,554,0):: CALL LOCATE
(#1,36,161)
1660 CALL SOUND(600,587,0):: CALL LOCATE
(#1,32,176)
1670 CALL SOUND(300,440,0):: CALL LOCATE
(#1,32,193)
1680 CALL SOUND(600,698,0):: CALL LOCATE
(#1,24,208)
1690 CALL SOUND(150,659,0):: CALL LOCATE
(#1,32,223)
1700 CALL SOUND(150,587,0):: CALL LOCATE
(#1,32,239)
1710 CALL SOUND(600,587,0):: CALL LOCATE
(#1,80,32)
1720 CALL SOUND(300,554,0):: CALL LOCATE
(#1,84,56)
1730 CALL SOUND(600,587,0):: CALL LOCATE
(#1,80,72)
1740 CALL SOUND(300,440,0,554,2):: CALL
LOCATE(#1,84,96,#2,32,41)
1750 CALL SOUND(600,587,0,294,3):: CALL
LOCATE(#1,80,112,#2,32,57)
1760 CALL SOUND(300,440,0,659,4):: CALL
LOCATE(#1,76,128,#2,32,70)
1770 CALL SOUND(600,349,0,698,7):: CALL
LOCATE(#1,72,152,#2,40,81)
1780 CALL SOUND(300,294,0,392,2):: CALL
LOCATE(#1,84,169,#2,48,97)
1790 CALL SOUND(600,440,2,349,0):: CALL
LOCATE(#1,88,185,#2,32,113)
1800 CALL SOUND(300,494,4,294,0):: CALL
LOCATE(#1,96,203,#2,28,129)
1810 CALL SOUND(600,523,2,440,0):: CALL
LOCATE(#1,80,222,#2,36,145)
1820 CALL SOUND(300,554,3,392,0):: CALL
LOCATE(#1,84,241,#2,36,161)
1830 CALL SOUND(600,587,3,349,0):: CALL
LOCATE(#1,136,33,#2,32,176)
1840 CALL SOUND(300,440,2,330,0):: CALL
LOCATE(#1,140,49,#2,32,193)
1850 CALL SOUND(600,698,8,294,0):: CALL
LOCATE(#1,144,65,#2,24,208)
1860 CALL SOUND(150,659,6,349,0):: CALL
LOCATE(#1,136,81,#2,32,223)
1870 CALL SOUND(150,587,3,349,0):: CALL
LOCATE(#2,32,239)
1880 CALL SOUND(600,587,2,440,0):: CALL
LOCATE(#1,128,97,#2,80,32)
1890 CALL SOUND(300,554,3,220,0):: CALL
LOCATE(#1,148,114,#2,84,56)
1900 CALL SOUND(600,587,4,294,0):: CALL
LOCATE(#1,144,134,#2,80,72)
1910 FOR X=1 TO 4
```

```
1920 CALL SOUND(300,440,0,554,2,659,8)::
 CALL LOCATE(#1,124,151,#2,84,96,#3,32,4
1)
1930 IF X=2 THEN CALL COLOR(#1,3,#2,5,#3
,9)
1940 IF X=3 THEN CALL COLOR(#1,5,#2,9,#3
,3)
1950 IF X=4 THEN CALL COLOR(#1,9,#2,3,#3
,5)
1960 CALL SOUND(600,587,0,294,3,698,6)::
 CALL LOCATE(#1,120,167,#2,80,112,#3,32,
57)
1970 CALL SOUND(300,440,0,659,6,554,2)::
 CALL LOCATE(#1,132,192,#2,76,128,#3,32,
70)
1980 CALL SOUND(450,349,0,698,8,587,3)::
 CALL LOCATE(#1,128,208,#2,72,152,#3,40,
81)
1990 CALL SOUND(150,349,0,698,8,659,6)::
 CALL LOCATE(#1,128,224)
2000 CALL SOUND(300,294,0,392,0,698,8)::
 CALL LOCATE(#1,120,240,#2,84,169,#3,48,
97)
2010 CALL SOUND(450,440,0,349,0,587,2)::
 CALL LOCATE(#1,176,32,#2,88,185,#3,32,1
13)
2020 CALL SOUND(150,440,0,349,0,554,2)::
 CALL LOCATE(#1,184,56)
2030 CALL SOUND(300,494,0,294,0,587,2)::
 CALL LOCATE(#1,176,72,#2,96,203,#3,28,1
29)
2040 CALL SOUND(600,523,0,440,0,659,6)::
 CALL LOCATE(#1,174,88,#2,80,222,#3,36,1
45)
2050 CALL SOUND(300,554,0,392,0,659,7)::
 CALL LOCATE(#1,174,104,#2,84,241,#3,36,
161)
2060 CALL SOUND(600,587,0,349,0,440,1)::
 CALL LOCATE(#1,176,118,#2,136,33,#3,32,
176)
2070 CALL SOUND(300,440,0,330,0,392,0)::
 CALL LOCATE(#1,180,129,#2,140,49,#3,32,
193)
2080 CALL SOUND(450,698,5,294,2,349,0)::
 CALL LOCATE(#1,184,145,#2,144,65,#3,24,
208)
2090 CALL SOUND(150,698,5,294,1,392,0)::
 CALL LOCATE(#1,180,161)
2100 CALL SOUND(150,659,4,349,0,440,1)::
 CALL LOCATE(#1,176,177,#2,136,81,#3,32,
223)
2110 CALL SOUND(150,587,1,349,0,440,0)::
 CALL LOCATE(#3,32,239)
2120 CALL SOUND(450,587,0,440,0,349,0)::
 CALL LOCATE(#1,184,193,#2,128,97,#3,80,
32)
2130 CALL SOUND(150,587,0,440,0,330,0)::
 CALL LOCATE(#1,188,209)
2140 CALL SOUND(300,587,0,294,0,147,5)::
 CALL LOCATE(#1,188,225,#2,148,113,#3,84
,56)
2150 IF X=4 THEN 2180
2160 CALL SOUND(600,587,0,294,0,147,5)::
 CALL LOCATE(#1,188,241,#2,144,134,#3,80
,72)
2170 NEXT X
2180 CALL SOUND(2000,587,0,294,0,147,5):
: CALL LOCATE(#1,188,241,#2,144,134,#3,8
0,72)
2190 FOR F=1 TO 1500 :: NEXT F
2200 CALL DELSPRITE(ALL):: GOTO 360
```

Chicken helper

Claude Lamy



Chicken Helper est un programme qui nous vient, lui aussi, du club TI-SOFT-HOME de Bruxelles. Il est basé sur le principe d'un jeu de bistrot (ou d'estaminet) maintenant assez connu : un personnage essaie de traverser une route très fréquentée et il veut, bien entendu, éviter de se faire écraser.

Le listing est très parlant, aussi ne fais-je ici qu'une très courte présentation. Vous constaterez que les représentations graphiques sont très claires. Notamment, la tache rouge laissée par le personnage qui vient de se faire "avoir" par une voiture est du plus bel effet.

Il y a cinq piétons qui, chacun à son tour, tentent la traversée. Lorsque l'un d'eux a réussi, il recommence dans l'autre sens, et ainsi de suite tant qu'il ne lui est rien arrivé. Les autres, pendant ce temps, attendent patiemment. Le jeu se termine lorsque le cinquième piéton est détruit. Le score est le nombre de traversées réussies, s'il y en a, car ce jeu réclame une très grande habileté.

```
50 REM ***************************
60 REM *  COPYRIGHT 99 MAGAZINE  *
70 REM *      ET PHIL BELJAEFF   *
80 REM ***************************
90 REM
100 REM
110 ON BREAK NEXT :: CALL DELSPRITE(ALL)
:: CALL CLEAR :: CALL MAGNIFY(3):: CALL
```

```
SCREEN(2):: RANDOMIZE :: RESTORE
120 DEF F=INT(RND*11)+20+XXX
130 FOR D=1 TO 8 :: CALL COLOR(D,16,1)::
 NEXT D :: CALL COLOR(0,15,15,13,6,1,14,
16,8):: N=9 :: L=120
140 DISPLAY AT(12,9)BEEP:"CHICKEN HELPER
" :: DISPLAY AT(14,8):"DE PHIL BELJAEFF"
150 DISPLAY AT(24,2):"APPUYEZ SUR UNE TO
UCHE"
160 CALL KEY(0,K,S):: IF S=0 THEN 160
170 CALL CLEAR :: DISPLAY AT(1,1):"OBJET
 : AIDER LES CHICKENS":"A TRAVERSER L'AU
TOROUTE":"EN UTILISANT 'E' & 'X'"
180 DISPLAY AT(4,1):"(FLECHES HAUT ET BA
S)."
190 DISPLAY AT(6,1):"VOTRE SCORE ET LE N
OMBRE DE ":"VIES PERDUES SERA AFFICHE":"
EN PERMANENCE.": :"A CHAQUE TRAVERSEE, V
OUS"
200 DISPLAY AT(11,1):"GAGNEREZ UNE CHANC
E DE":"SURVIE SUPPLEMENTAIRE."
210 DISPLAY AT(24,1):"UNE TOUCHE POUR CO
MMENCER"
220 CALL KEY(0,K,S):: IF S=0 THEN 220
230 CALL CLEAR
240 B$="00000000808818E0" :: C$="0000000
001010001" :: D$=RPT$("0",16)
250 CALL CHAR(96,"0000000F0F000000"&D$&"
000020F0F0200000"&D$)
260 CALL CHAR(100,"0000040F0F040000"&D$&
"000000F0F0000000"&D$)
270 CALL CHAR(104,"00187F7F7F7F1800"&D$&
"00C0F0F8F8F0C000"&D$)
280 CALL CHAR(108,"000C3F7F7F3F0C00"&D$&
"0060F8F8F8F86000"&D$)
290 CALL CHAR(112,"18FFFFFFFFFFFF18"&D$&
"60FCFEFEFEFEFC60"&D$)
300 CALL CHAR(116,"0C7FFFFFFFFF7F0C"&D$&
"30FEFEFEFEFEFE30"&D$)
310 CALL CHAR(120,C$&"0105020000000000"&
B$&"80C0203000000000")
320 CALL CHAR(124,C$&"0103040C00000000"&
B$&"80A0400000000000")
330 CALL CHAR(128,C$&"0301010700000000"&
B$&"808080E000000000")
340 CALL CHAR(132,"181800183C18183C")::
CALL CHAR(136,"FF00000000000000F0000000000
00000FF"&D$)
350 CALL CHAR(140,"0000020707030F1F3F3F1
F0F3F7B30000020B0F0E0F0FCF0E0F0F8FCFCF8F
830")
360 CALL HCHAR(5,1,31,64):: CALL HCHAR(2
1,1,31,64)
370 FOR D=7 TO 19 STEP 2 :: CALL HCHAR(D
,1,136,32):: CALL HCHAR(D+1,1,138,32)::
NEXT D :: CALL HCHAR(20,1,137,32)
380 DISPLAY AT(2,10):"SCORE:    0" :: DIS
PLAY AT(3,1):RPT$(CHR$(132),9)
390 DATA 96,100,104,116,96,100,112,116,1
12,100,104,108,104,116
400 FOR D=1 TO 7
410 READ Q,R :: CALL SPRITE(#D,Q,2,(D+6)
*8-7,1,0,F,#D+7,R,2,(D+13)*8-7,1,0,-F)::
 NEXT D
420 CALL SPRITE(#15,128,5,33,121)
430 CALL KEY(0,K,S)
440 IF K=69 THEN A=-9 :: GOTO 460
450 IF K=88 THEN A=9 ELSE A=0 :: GOTO 49
0
460 CALL POSITION(#15,Y,X):: CALL COINC(
ALL,Z):: IF Z<0 THEN GOSUB 610 :: GOTO 4
30
470 IF Y<33 THEN A=0 :: CALL LOCATE(#15,
33,121):: IF W=1 THEN GOSUB 540
480 IF Y>161 THEN A=0 :: CALL LOCATE(#15
,161,121):: IF W=0 THEN GOSUB 540
490 CALL MOTION(#15,A,0)
500 CALL COINC(ALL,Z):: IF Z<0 THEN GOSU
B 610
510 IF L=120 THEN L=124 ELSE L=120
520 CALL PATTERN(#15,L)
530 GOTO 430
540 CALL MOTION(#15,0,0)
550 CALL SOUND(300,415,0):: CALL SOUND(1
00,110,30):: CALL SOUND(300,349,0):: CAL
L SOUND(100,110,30):: CALL SOUND(1000,27
7,0)
560 XXX=XXX+2 :: XXX=MIN(XXX,50):: W=1-W
 :: SC=SC+1 :: N=N+1
570 CALL PATTERN(#15,128):: IF N>28 THEN
 Q=4 :: R=1 :: GOTO 580 :: ELSE Q=3 :: R
=0
580 DISPLAY AT(2,16):USING "####":SC ::
DISPLAY AT(Q,1):RPT$(CHR$(132),N-(R*28))
590 FOR E=1 TO 7 :: CALL MOTION(#E,0,F,#
E+7,0,-F):: NEXT E
600 RETURN
610 CALL MOTION(#15,0,0)
620 N=N-1 :: IF N>28 THEN Q=4 :: R=1 ::
GOTO 630 :: ELSE Q=3 :: R=0
630 DISPLAY AT(Q,1):RPT$(CHR$(132),N-(R*
28))
640 CALL COLOR(#15,9):: CALL PATTERN(#15
,140)
650 CALL SOUND(300,220,0):: CALL SOUND(1
00,220,30):: CALL SOUND(150,220,0):: CAL
L SOUND(100,220,30):: CALL SOUND(2000,29
4,0)
660 CALL SOUND(1,110,30)
670 IF W=0 THEN CALL LOCATE(#15,33,121)E
LSE CALL LOCATE(#15,161,121)
680 CALL COLOR(#15,5)
690 CALL PATTERN(#15,128):: IF N>0 THEN
RETURN
700 DISPLAY AT(23,10):"TERMINE" :: DISPL
AY AT(24,10):"FAITES 'REDO'"
710 CALL KEY(0,K,S):: IF K<>6 THEN 710 E
LSE RUN
```

La sauvegarde des fichiers et programmes

Jean Sahal

SAVE CS1 : c'est la commande familière à tous les utilisateurs du TI-99 qui travaillent avec un petit budget. Sans doute considèrent-ils que l'usage non professionnel de leur ordinateur ne justifie pas la dépense importante que représente l'achat du système de disquettes. Ce dernier est bien le plus pratique sous tous les rapports. Mais le magnétophone à cassettes reste pratiquement l'unique moyen de sauvegarder le précieux résultat de longues heures de travail. C'est cependant un procédé délicat qui réserve des déceptions inattendues, d'autant plus amères souvent que l'on a scrupuleusement respecté la marche à suivre.

Les lignes qui suivent peuvent vous aider à éviter les pièges de l'utilisation du magnétophone à cassettes. Il ne s'agit pas, évidemment, de paraphraser le mode d'emploi tel qu'il est écrit dans le manuel du TI-99 (qu'il est pourtant conseillé de lire attentivement !), mais de compléter ces instructions par quelques remarques inspirées par la pratique de ce système.

Nul n'est à l'abri d'incidents de parcours, même l'amateur le plus chevronné travaillant avec un magnétophone neuf qu'il consacre exclusivement à cet usage. Il faut comprendre qu'un programme est une chose fragile, faite de signaux traduisant des nombres binaires, eux-mêmes représentant des signes alphanumériques ou spéciaux : chiffres, lettres, ponctuation, etc. Il suffit d'un tout petit incident de transmission pour que le programme soit "démoli" et que le message "ERROR DETECTED IN DATA" apparaisse. Parmi les causes d'incidents les plus fréquentes, on notera une installation électrique vétuste génératrice de "faux contacts" produisant des parasites, ou des potentiomètres (volume et tonalité) mal réglés; bref, tout ce qui empêche l'ordinateur de "percevoir" les signaux de manière absolument parfaite.

Le magnétophone à cassettes

Si vous possédez déjà un magnétophone et si vous voulez l'utiliser pour votre ordinateur, il pourra faire l'affaire, à condition toutefois qu'il soit pourvu des fiches "Jack" femelles, l'une pour la sortie écouteur (pour envoyer les programmes vers l'ordinateur), et l'autre pour l'entrée micro (pour enregistrer les programmes). La prise de télécommande et le compteur ne sont pas indispensables : on peut aisément suppléer à leur absence par une attention accrue. Il est dans ce cas vivement conseillé d'utiliser le "truc" exposé dans le numéro 1 de "99 Magazine" à la page 65, consistant à faire précéder vos programmes sur la cassette par une annonce sonore comportant un compte à rebours. Il est également sage de "suivre" auditivement le transfert du programme dans le haut-parleur du téléviseur : cela permet de détecter rapidement une éventuelle anomalie et peut permettre d'en déterminer la cause.

On remarquera que tout programme est précédé d'un "blanc" laissé par l'ordinateur lui-même au moment de l'enregistrement, durant environ quatorze secondes. Ce blanc est suivi d'un sifflement d'une durée de quatre secondes. Si le magnétophone ne comporte pas de télécommande, c'est pendant ce sifflement qu'il faut appuyer sur la touche "ENTER" de l'ordinateur. Cette précaution est bien entendu inutile lorsque l'on dispose de la télécommande car le défilement de la bande est commandé directement par l'ordinateur.

Remarque pour ceux qui disposent de la télécommande et qui utilisent des annonces sonores : ils doivent lire l'annonce sonore avant d'écrire "OLD CS1" sur l'ordinateur. Ainsi, la télécommande est libre et on peut faire tourner le magnétophone. Ce n'est que lorsque l'on a entendu entièrement l'annonce (on se trouve alors sur le "blanc" de quatorze secondes) que l'on arrête le magnétophone et que l'on commence la procédure normale "OLD CS1". Si on fait cette commande avant l'annonce sonore, l'ordinateur va lire cette dernière et y chercher des signaux correspondant à des signaux de programmes ou de données. Comme il ne les trouvera pas, il affichera son message d'erreur...

Quelque soit le modèle de magnétophone utilisé, il est tout à fait préférable de l'affecter uniquement à l'usage de l'ordinateur. En ce qui concerne les potentiomètres, il faut régler le volume au maximum et la tonalité au plus aigu. Lorsque le système aura fait ses preuves, il vaudra mieux ne plus y toucher pour ne pas risquer de modifier les réglages effectués. De plus, on risque, à force de débrancher et rebrancher sans cesse, de détériorer les fiches qui ne donneraient plus alors des contacts d'aussi bonne qualité qu'il le faidrait.

Pour ceux qui ne sont pas encore équipés, ou qui souhaitent acquérir un nouveau magnétophone pour leur TI-99, signalons la sortie récente, chez Texas Instruments, d'un magnétophone à cassettes spécialement conçu pour la sauvegarde des fichiers de l'ordinateur. La qualité des transmissions à partir de ce matériel est en principe irréprochable et générera certainement moins souvent de message d'erreur qu'avec tout autre matériel.

Le cordon

Les "initiés" désignent le cordon sous le nom de "Interface Cassettes". Il en existe deux modèles sur le marché. Le plus ancien est conçu pour relier deux magnétophones à l'ordinateur. Il peut se révéler très utile pour ceux qui utilisent ce support magnétique pour faire de la gestion de fichier : on lit un fichier de manière séquentielle sur l'un des magnétophones, on traite les informations et on stocke le fichier mis à jour sur le second magnétophone. De ce point de vue, le câble à deux dérivations est indispensable. Il présente cependant un défaut : il ne comporte de télécommande que pour l'unité CS1. De la sorte, lors de telles opérations, le manipulateur est contraint de déclencher lui-même les avances et arrêts de la bande réceptrice.

Pour ceux qui n'utilisent qu'un seul magnétophone, ne coupez pas le second fil : il pourra vous servir si, comme nous l'avons vu plus haut, vous voulez faire de la gestion de fichiers. Prenez simplement la précaution d'isoler les deux fiches que vous n'utilisez pas de manière à éviter qu'elles ne provoquent des courts-circuits. Pour celles que vous branchez, il est conseillé de les consolider pour ne pas risquer de dessouder les fils lors d'une opération de débranchement du magnétophone.

Le nouveau câble désormais commercialisé est conçu, lui, pour un seul magnétophone. Il est plus fonctionnel que son prédécesseur et les

fiches en sont nettement plus solides car revêtues d'un moulage en caoutchouc. La dérivation CS2 n'existant plus, il n'y a plus de fil qui traîne inutilement.

La cassette

Si l'ensemble du système ne comporte aucune faiblesse, des cassettes très bon marché peuvent suffire. Elles sont même préférables car, vu leur faible coût, on n'hésitera pas à les découper, par exemple pour mettre un seul programme par face. Le supplément de bande sera purement et simplement jeté, mais il faut prendre une précaution : ne couper qu'à la longueur du plus long des deux programmes enregistrés. Une certaine gymnastique est nécessaire dans laquelle l'habitude donne une très grande habileté.

D'autres précautions doivent être prises : ces cassettes très économiques ont une couche d'enduit magnétique particulièrement mince qui peut se révéler à la longue insuffisante pour la conservation de programmes informatiques. Il faut donc dupliquer toutes les sauvegardes, c'est-à-dire les faire en double, en contrôlant à chaque fois la validité de l'enregistrement. Cela risque évidemment d'alourdir considérablement les procédures de mise au point d'un programme, mais c'est quand même préférable que de perdre le fruit de longues heures de travail intense.

Vous pouvez aussi décider de n'utiliser que des cassettes de bonne qualité : les cassettes HI-FI LOW-NOISE sont les meilleures que vous pourrez trouver. Dans ce cas, s'il y a des incidents, ils ne peuvent évidemment pas provenir de la qualité de la bande mais, plus probablement, d'une erreur de manipulation. Si elles ont, précisément, l'avantage de leur qualité, elles ont en revanche l'inconvénient de leur prix élevé, ce qui fait que l'on hésitera certainement à les découper et encore plus à en jeter des morceaux. De la sorte, si un programme se trouve en fin de bande, il faudra opérer un déroulement complet pour le trouver, ce qui constitue incontestablement une perte de temps. On pourra pallier cet inconvénient en mettant en début de cassette les programmes les plus utilisés.

Quelques trucs

Prenez la précaution de sauvegarder vos programmes en deux endroits différents. En particulier si vous utilisez fréquemment l'un d'entre eux, faites deux sauvegardes en plus de la bande de travail : il se trouvera bien quelqu'un chez vous pour en perdre ou détruire une. Si, dans le même temps, vous "écrasez" votre bande de travail...

Evitez de programmer le week-end quand votre conjoint fait fonctionner des appareils électriques. Si les plombs sautent, vous pouvez dire adieu à votre programme. Si vraiment vous ne pouvez vous en passer annoncez alors que vous emménerez ensuite votre famille au restaurant. Personne n'aura de choses urgentes à faire et vous serez tranquille pendant quelques heures (voilà qui est bien fait pour la réconciliation des familles !).

Signalons enfin l'existence d'un module enfichable très pratique : la Mini-Mémoire qui permet de stocker des programmes d'une taille maximale de quatre kilo-octets, instantanément et sans le secours d'aucun périphérique. Ce module renferme une pile censée durer deux ans qui assure la conservation même lorsque l'ordinateur est débranché ou lorsque l'on retire le module de son logement. C'est très utile pour ceux qui sont fréquemment obligés d'interrompre leur travail et qui ne disposent pas de beaucoup de temps ou d'espace pour s'organiser. En outre, ce module permet de s'initier à la programmation en assembleur, à condition toutefois de pouvoir se contenter de la très faible documentation qui l'accompagne.

Dernière chose importante : ne confondez pas OLD et SAVE. Surtout, ne tapez pas OLD en pensant SAVE car tout serait détruit. C'est là que le mot SAVE prend toute la signification qu'il a dans l'expression S.O.S. (Save Our Souls : sauvez nos âmes !).

Le Mastermind

Edouard Jouan

Le programme que je vous présente ici s'inspire du célèbre "Mastermind" et provient du club TISOFT-HOME de Bruxelles. L'ordinateur choisit parmi cinq couleurs un arrangement de quatre d'entre elles. Le joueur doit deviner la nature de cet arrangement en faisant 9 propositions successives (au maximum). En réponse à chacune de ces propositions, le programme indique combien de couleurs sont bonnes mais mal disposées et combien sont bonnes et en bonne place.

Il existe une méthode infaillible de résolution du problème qui aboutit en général à la bonne combinaison après cinq propositions. Mais ne comptez pas que je vous la donne dans ces lignes : cela supprimerait tout l'intérêt du jeu et ce serait bien dommage. A vous donc de la découvrir, ce qui suppose, bien entendu, que vous ayez auparavant tapé le programme (ou que vous en disposiez sur l'une des merveilleuses cassettes d'accompagnement de "99" !).

Comme tout programme de jeu correctement écrit, celui-ci débute par un énoncé des règles du jeu (lignes 240 à 430). Remarquez la structure de la ligne 430 : le programme vous demande de faire ENTER pour continuer et attend la valeur d'une variable A$. Si on fait simplement ENTER comme demandé, cette variable sera vide car on n'aura pas indiqué de valeur. Cela n'a aucune importance, car l'objet de cette ligne est simplement de mettre le programme en attente au moyen de l'instruction INPUT et la variable A$ n'est utilisée nulle part ailleurs.

Si vous disposez du module de Basic Etendu, vous pouvez faire tourner ce programme dans ce langage : il n'utilise en effet pas de caractère au-delà du caractère 143. Vous pouvez alors compacter les lignes 460 à 590 en une seule ligne :

460 J=0::F=15::E=1

Les lignes 600 à 1050 fabriquent les caractères servant à construire les graphiques. Remarquez la ligne 600

qui renvoie à la ligne 1060 si J=1. Or la ligne 460 affecte la valeur 0 à cette variable. Lors du lancement du programme, la valeur de J étant 0, les lignes 610 à 1050 sont exécutées. La ligne 610 affecte la valeur 1 à la variable, J de sorte qu'à tout nouveau passage par la ligne 600, le programme sautera directement à la ligne 1060 : il n'est en effet pas nécessaire, lorsque l'on décide de rejouer à ce jeu, de construire une deuxième fois les mêmes caractères.

Le choix des couleurs à deviner commence à la ligne 650 : A=INT(5★RND)★8 La valeur de A est donc choisie entre 0, 8, 16 et 32. Il faut en effet se souvenir que les codes ASCII utilisés par le TI-99 sont groupés par séries de huit. Lorsque l'on affecte une couleur à un caractère, on l'affecte en réalité à tous les caractères du groupe dont il fait partie. S'agissant ici de déterminer quatre caractères de couleurs différentes, il faut que chacun d'eux soit compris dans un groupe différent des autres. La ligne 1090 du programme construit le caractère qui servira, à la fin du jeu, à indiquer la bonne réponse en ce qui concerne la première couleur. Ce caractère est celui de numéro 96, 104, 112 ou 120, c'est-à-dire qu'il appartient à l'un des quatre groupes dans lesquels ces numéros de caractères sont compris. Les couleurs de ces quatre groupes ont été préalablement déterminées dans les lignes 760 à 790.

Il y a plusieurs fois dans le programme l'instruction CALL KEY. Je ne vous ferai pas l'injure de vous en rappeler la signification : vous devez maintenant la connaître. Si, cependant, vous avez encore quelques difficultés à son sujet, vous pouvez vous reporter à l'article de Bernard Braun dans le numéro 1 de "99 Magazine".

Si vous frappez "S" avant d'avoir fait ENTER, vous annulez purement et simplement la proposition que vous avez commencé à faire. Si, en revanche, vous frappez "M", le jeu se termine avec indication de la solution.

Vous remarquerez à l'usage que, si vous laissez votre doigt appuyé trop longtemps sur l'initiale d'une couleur, cette dernière se positionne deux fois dans votre proposition, ce qui est gênant. Il vous appartient de compléter ce programme afin de vous protéger contre ce genre d'erreur. Vous pouvez par exemple inclure les lignes suivantes :

1054 CALL KEY(0,ARRET,S)

1058 IF S=−1 then 1054

1059 S=0

De la sorte, après le CALL KEY de la ligne 1050, le programme est interrompu tant que votre doigt n'est pas relevé et la couleur que vous



avez indiquée ne peut être marquée deux fois (ce truc est donné par Denise Amrouche dans son programme du Jeu de l'Alphabet dans le numéro 1 de "99 Magazine").

Le programme d'origine était en anglais ; les couleurs sont dénommées "red, white, blue, yellow, green". Dans le but de vous laisser travailler un peu (!), j'ai laissé ces dénominations telles qu'elles étaient. Si vous le voulez, vous pourrez les traduire, mais une difficulté se présente : "white" se traduit par "blanc" qui a la même initiale que "bleu". Il faut alors soit désigner le bleu par un autre nom (cyan ou ciel...), soit remplacer l'une de ces deux couleurs par une autre dont l'initiale ne présentera plus de problème. Il ne faudra pas oublier de modifier en conséquence soit la ligne 1280, soit... Mais il n'y a pas de test "IF KEY=66" pour savoir si le "B" a été frappé. En effet, si l'on n'a pas frappé l'une des autres touches demandées, on est censé avoir frappé le "B". Vous ne pourrez donc modifier que la couleur blanche.

La manière dont sont analysées les propositions faites par le joueur me semblent assez évidentes à la lecture du programme. Je ne vais donc pas la répéter ici puisque, tout en l'entrant dans votre machine, vous pourrez la comprendre. Remarquez cependant la ligne 2500 : elle remet à zéro toute une série de variables d'un seul coup. J'ai essayé : on peut en mettre autant que la ligne peut en contenir (en fonction de la dénomination des variables que l'on utilise). Remarquez aussi la ligne 2550 : la question "ENCORE? (O/N)" est suivie de CHR$(30) qui est le curseur. Lorsque l'on demande par CALL KEY l'introduction d'une valeur, l'ordinateur ne dispose en principe pas de curseur car le caractère frappé ne sera pas affiché. La présence de ce curseur clarifie cependant une question et il n'est pas toujours inutile de le faire apparaître.

Ce programme est tout à fait perfectible, ne serait-ce que pour modifier le graphisme s'il n'était pas à votre goût. On constate par exemple la présence d'instructions CALL KEY à divers niveaux qui semblent bien pouvoir être remplacées par une seule disposée différemment. Il faudrait alors revoir en profondeur la structure du programme. Ceci aurait cependant un inconvénient important : celui de ralentir l'exécution. Il vaut donc mieux le laisser tel qu'il est, même si, à la lecture, il semble un peu bizarre. Il n'y a donc plus maintenant qu'à passer de longs et délicieux moments pour découvrir la méthode systématique de résolution de ce jeu. Essayez quand même de ne pas attraper de migraine...

```
100 REM ******************************
110 REM *  COPYRIGHT 99 MAGAZINE     *
120 REM *  ET TISOFT HOME            *
130 REM ******************************
140 REM
150 CALL CLEAR
160 PRINT " JEU DES 4 COULEURS"
170 FOR TH=1 TO 12
180 PRINT
190 NEXT TH
200 FOR DELAY=1 TO 200
210 NEXT DELAY
220 CALL CLEAR
230 PRINT "INSTRUCTIONS?(O/N) "
240 CALL KEY(0,KEY,STAT)
250 IF KEY=79 THEN 280
260 IF KEY=78 THEN 440
270 GOTO 240
280 CALL CLEAR
290 PRINT "        INSTRUCTIONS"
300 PRINT
310 PRINT "'M' DONNE LA SOLUTION MAIS  L
```

```
E JEU S'ARRETE."
320 PRINT
330 PRINT
340 PRINT "ESSAYEZ DE RECONSTITUER LE  G
ROUPE DES 4 COULEURS EN 9  ESSAIS OU MOI
NS."
350 PRINT
360 PRINT
370 PRINT "  X=BON ET EN BONNE PLACE,
 /=BON, MAIS MAL PLACE"
380 PRINT "X ET / VOUS INDIQUENT LE     N
OMBRE DE COULEURS BIEN     CHOISIES."
390 PRINT
400 PRINT "'S' VOUS PERMET DE MODIFIER V
OTRE CHOIX AVANT DE FAIRE  'ENTER'."
410 PRINT
420 PRINT
430 INPUT "FAITES ENTER POUR CONTINUER":
A$
440 CALL CLEAR
450 CALL SCREEN(14)
460 J=0
470 F=15
480 E=1
490 G=7
500 XA=0
510 XB=0
520 XC=0
530 XD=0
540 WA=0
550 WB=0
560 WC=0
570 WD=0
580 Z=0
590 Y=0
600 IF J=1 THEN 1060
610 J=1
620 RANDOMIZE
630 CALL CHAR(96,"FFFFFFFFFFFFFFFF")
640 CALL CHAR(104,"FFFFFFFFFFFFFFFF")
650 A=INT(5*RND)*8
660 CALL CHAR(112,"FFFFFFFFFFFFFFFF")
670 CALL CHAR(120,"FFFFFFFFFFFFFFFF")
680 B=INT(5*RND)*8
690 IF B=A THEN 680
700 CALL CHAR(128,"FFFFFFFFFFFFFFFF")
710 CALL CHAR(136,"FFFFFFFFFFFFFFFF")
720 CALL COLOR(9,4,1)
730 C=INT(5*RND)*8
740 IF C=B THEN 730
750 IF C=A THEN 730
760 CALL COLOR(10,6,1)
770 CALL COLOR(11,10,1)
780 CALL COLOR(12,12,1)
790 CALL COLOR(13,16,1)
800 D=INT(5*RND)*8
810 IF D=A THEN 800
820 IF D=B THEN 800
830 IF D=C THEN 800
840 CALL HCHAR(1,10,52)
850 CALL HCHAR(1,12,67)
860 CALL HCHAR(1,13,79)
870 CALL HCHAR(1,14,85)
880 CALL HCHAR(1,15,76)
890 CALL HCHAR(1,16,69)
900 CALL HCHAR(1,17,85)
910 CALL HCHAR(1,18,82)
920 CALL HCHAR(1,19,83)
930 CALL HCHAR(7,2,82)
940 CALL HCHAR(9,2,87)
950 CALL HCHAR(11,2,66)
960 CALL HCHAR(13,2,89)
970 CALL HCHAR(15,2,71)
980 FOR TH=7 TO 15 STEP 2
990 CALL HCHAR(TH,3,61)
1000 NEXT TH
1010 CALL HCHAR(7,4,112)
1020 CALL HCHAR(9,4,128)
1030 CALL HCHAR(11,4,104)
1040 CALL HCHAR(13,4,120)
1050 CALL HCHAR(15,4,96)
1060 CALL HCHAR(4,15,136,4)
1070 CALL KEY(0,KEY,STY)
1080 IF KEY=77 THEN 1090 ELSE 1160
1090 CALL HCHAR(4,15,96+A)
1100 CALL HCHAR(4,16,96+B)
1110 CALL HCHAR(4,17,96+C)
1120 CALL HCHAR(4,18,96+D)
1130 FOR DELAY=1 TO 200
1140 NEXT DELAY
1150 GOTO 2550
1160 FOR GH=6 TO 24 STEP 2
1170 CALL HCHAR(GH,14,136,6)
1180 NEXT GH
1190 CALL VCHAR(6,14,136,18)
1200 CALL VCHAR(6,19,136,18)
1210 CALL KEY(0,KEY,STY)
1220 IF KEY=77 THEN 1090
1230 CALL KEY(0,KEY,STY)
1240 IF KEY=83 THEN 1250 ELSE 1280
1250 CALL HCHAR(G,15,CLEAR,4)
1260 F=15
1270 E=1
1280 IF KEY=87 THEN 1290 ELSE 1340
1290 S(E)=32
1300 CALL HCHAR(G,F,128)
1310 F=F+1
1320 E=E+1
1330 IF E=5 THEN 1620
1340 IF KEY=82 THEN 1350 ELSE 1410
1350 S(E)=16
1360 CALL HCHAR(G,F,112)
1370 F=F+1
1380 E=E+1
1390 IF E=5 THEN 1620
1400 REM  POUR LES 99/4 : DISPOSEZ ICI L
A LIGNE 1500
1410 IF KEY=66 THEN 1420 ELSE 1470
```

```
1420 S(E)=8
1430 CALL HCHAR(G,F,104)
1440 F=F+1
1450 E=E+1
1460 IF E=5 THEN 1620
1470 IF KEY=71 THEN 1480 ELSE 1530
1480 S(E)=0
1490 CALL HCHAR(G,F,96)
1500 F=F+1
1510 E=E+1
1520 IF E=5 THEN 1620
1530 REM  N'EFFACEZ PAS CETTE LIGNE !!!
1540 CALL KEY(0,KEY,STAT)
1550 IF KEY=89 THEN 1560 ELSE 1210
1560 S(E)=24
1570 CALL HCHAR(G,F,120)
1580 F=F+1
1590 E=E+1
1600 IF E=5 THEN 1620
1610 GOTO 1210
1620 IF A=S(1)THEN 1630 ELSE 1650
1630 Y=Y+1
1640 XA=1
1650 IF B=S(2)THEN 1660 ELSE 1680
1660 Y=Y+1
1670 XB=1
1680 IF C=S(3)THEN 1690 ELSE 1710
1690 Y=Y+1
1700 XC=1
1710 IF D=S(4)THEN 1720 ELSE 1740
1720 Y=Y+1
1730 XD=1
1740 IF XA=1 THEN 1890
1750 IF XB=1 THEN 1800
1760 IF A=S(2)THEN 1770 ELSE 1800
1770 Z=Z+1
1780 WB=1
1790 GOTO 1890
1800 IF XC=1 THEN 1850
1810 IF A=S(3)THEN 1820 ELSE 1850
1820 Z=Z+1
1830 WC=1
1840 GOTO 1890
1850 IF XD=1 THEN 1890
1860 IF A=S(4)THEN 1870 ELSE 1890
1870 Z=Z+1
1880 WD=1
1890 IF XB=1 THEN 2060
1900 IF XA=1 THEN 1950
1910 IF B=S(1)THEN 1920 ELSE 1950
1920 Z=Z+1
1930 WA=1
1940 GOTO 2060
1950 IF XC=1 THEN 2010
1960 IF WC=1 THEN 2010
1970 IF B=S(3)THEN 1980 ELSE 2010
1980 Z=Z+1
1990 WC=1
2000 GOTO 2060
2010 IF XD=1 THEN 2060
2020 IF WD=1 THEN 2060
2030 IF B=S(4)THEN 2040 ELSE 2060
2040 Z=Z+1
2050 WD=1
2060 IF XC=1 THEN 2240
2070 IF XA=1 THEN 2130
2080 IF WA=1 THEN 2130
2090 IF C=S(1)THEN 2100 ELSE 2130
2100 Z=Z+1
2110 WA=1
2120 GOTO 2240
2130 IF XB=1 THEN 2190
2140 IF WB=1 THEN 2190
2150 IF C=S(2)THEN 2160 ELSE 2190
2160 Z=Z+1
2170 WB=1
2180 GOTO 2240
2190 IF XD=1 THEN 2240
2200 IF WD=1 THEN 2240
2210 IF C=S(4)THEN 2220 ELSE 2240
2220 Z=Z+1
2230 WD=1
2240 IF XD=1 THEN 2420
2250 IF XA=1 THEN 2310
2260 IF WA=1 THEN 2310
2270 IF D=S(1)THEN 2280 ELSE 2310
2280 Z=Z+1
2290 WA=1
2300 GOTO 2420
2310 IF XB=1 THEN 2370
2320 IF WB=1 THEN 2370
2330 IF D=S(2)THEN 2340 ELSE 2370
2340 Z=Z+1
2350 WB=1
2360 GOTO 2420
2370 IF XC=1 THEN 2420
2380 IF WC=1 THEN 2420
2390 IF D=S(3)THEN 2400 ELSE 2420
2400 Z=Z+1
2410 WC=1
2420 IF Y=0 THEN 2440
2430 CALL HCHAR(G,10,88,Y)
2440 IF Z=0 THEN 2460
2450 CALL HCHAR(G,10+Y,47,Z)
2460 G=G+2
2470 IF Y=4 THEN 1090
2480 Y=0
2490 Z=0
2500 WA=WB=WC=WD=XA=XB=XC=XD=0
2510 E=1
2520 F=15
2530 IF G=25 THEN 1090
2540 GOTO 500
2550 PRINT "ENCORE? (O/N)";CHR$(30)
2560 CALL KEY(0,KEY,STAT)
2570 IF STAT=0 THEN 2560
2580 IF KEY=79 THEN 440
2590 END
```

Othello

Didier Bicking

Othello est le type même du jeu conçu pour être joué sur ordinateur. C'est pourquoi je vous présente ici ce programme qui nous vient de nos amis belges du club TI-SOFT HOME. Il présente l'avantage d'être écrit en Basic simple, ce qui le rend accessible même à ceux qui ne disposent que de la version de base du TI-99 : console et magnétophone à cassettes.

Ce programme n'est pas excessivement long : il ne comporte que 455 lignes. Mais il est construit de façon suffisamment astucieuse pour être très performant.



Avant tout, je vous rappelle les règles du jeu d'Othello. On dispose d'un damier de huit cases sur huit. Les pions des joueurs sont d'une couleur différente sur chaque face. Ici, ils sont rouges d'un côté et jaunes de l'autre. Les deux joueurs posent leurs pions toujours sur le même côté : l'un pose des pions jaunes, l'autre des pions rouges. Ils commencent par occuper les quatre cases du milieu du damier (en huit sur huit, c'est d'ailleurs plutôt un échiquier !). Ils posent ensuite leurs pions où ils le souhaitent, à la seule condition de prendre à chaque fois des pions à l'adversaire.

Les pions de l'adversaire sont pris lorsqu'ils sont, du fait de la pose d'un nouveau pion, disposés entre deux pions du joueur, que ce soit en ligne horizontale, verticale ou diagonale, qu'il s'agisse d'un seul pion ou de plusieurs sur un même alignement. La prise est constatée par le retournement. Ainsi, si le joueur rouge prend un pion jaune, il retourne ce dernier qui devient alors un pion rouge.

Lorsqu'aucune prise n'est possible, le joueur dont c'est le tour est alors autorisé à jouer n'importe où sur l'échiquier (NDLR : *la règle précise d'Othello stipule que, dans ce cas, le joueur doit passer.*). Le jeu se termine lorsque toutes les cases sont occupées ; le gagnant est celui qui a le plus de pions dans sa couleur.

Le programme autorise le jeu à deux joueurs. Il se contente alors d'enregistrer les coups, tout en vérifiant leur validité, et d'opérer les retournements occasionnés par chaque prise.

Il autorise aussi le jeu à un seul joueur contre le TI-99. Il possède donc une stratégie de jeu qui lui est propre et dont, à l'usage, vous comprendrez très vite la philosophie.

Une particularité doit être signalée en ce qui concerne l'exposé des règles du jeu par l'ordinateur : celui-ci est obtenu par listage des lignes 1000 à 1420 du programme. L'auteur a trouvé ce moyen pour économiser le temps d'apparition du texte lors du lancement du programme. Une fois que vous connaitrez suffisamment ces règles, vous pourrez en conséquence supprimer simplement ces lignes du programme. Voire même, vous pouvez tout bonnement vous dispenser de les taper, vous réservant la possibilité de pouvoir les lire dans ces pages (à condition de conserver précieusement cette excellente revue...)

Je ne vais pas, cette fois-ci, vous commenter le programme, mais plutôt vous exposer la stratégie à utiliser pour être à peu près sûr de gagner.

1 - Le principe même du jeu consiste à éviter que les pions que l'on pose puissent se trouver entre deux pions de l'adversaire. Pour cette raison, il est essentiel de s'assurer la maîtrise des côtés et aussitôt que possible de prendre les coins.

2 - Il faut ensuite "appuyer" son jeu sur ces pions dont la prise est, sinon impossible(ce qui est le cas des quatre coins), du moins très difficile à court terme.

3 - Lorsqu'un pion de l'adversaire se trouve au milieu d'une zone que l'on occupe en presque totalité, il faut tout faire pour le prendre car il est trop menaçant : l'adversaire peut en effet l'utiliser pour appuyer des coups dont l'objet est de briser votre domination dans la zone concernée.

4 - Les pions se trouvant dans les coins sont impossibles à reprendre, de même que ceux qui se trouvent sur les côtés et sont directement appuyés sur un coin. Il faut en conséquence, aussitôt que l'on a pris un coin, appuyer son jeu sur les deux côtés adjacents en développant une occupation du "terrain" autour du coin de départ.

Il m'est arrivé de faire des parties contre le TI-99 et d'arriver à ce qu'il ne lui reste plus de pion dans sa couleur. Le jeu s'arrête alors par ma victoire. Je ne me considère pas, loin de là, comme d'une intelligence supérieure. Il doit donc être possible à tout le monde d'en faire autant. Essayez! Je vous souhaite de bonnes et longues heures avec ce jeu passionnant.

Othello - TI Basic

```
1000 REM ****************************
1010 REM *   OTHELLO                 *
1020 REM *  COPYRIGHT 99 MAGAZINE    *
1030 REM *      ET J. CRAWFORD       *
1040 REM ****************************
1050 REM
1060 REM
1070 REM ECRIT POUR L'ORDINATEUR
1080 REM TI-99/4 OU TI-99/4A
1090 REM ce programme permet à deux joue
urs de jouer
1100 REM ensemble ou à un seul de jouer
contre
1110 REM l'ordinateur. Il y a 3 niveaux
de difficulté.
1120 REM Il faut avoir joué au moins 25
fois avant de pratiquer le niveau 3
1130 REM du moins pour les enfants de mo
ins de 13 ans.
1140 REM Le TI-99 met environ 30 seconde
s pour jouer.
1150 REM Au niveau 3, de nombreux joueur
s mettent plus longtemps!
1160 REM Le niveau 1 est un niveau de dé
monstration.
1170 REM
1180 REM Les mouvements sont entrés par
indication de la colonne puis de la lign
e.
1190 REM Toutes les autres opérations so
nt réalisées automatiquement.
1200 REM
1210 REM
1220 REM Règles :
1230 REM 1) On occupe en premier les 4 c
arrés du centre.
1240 REM 2) Après cela, chaque joueur jo
ue comme il l'entend
1250 REM  à condition toutefois que, à c
haque mouvement
1260 REM il capture au moins un carré de
 l'adversaire.
1270 REM 3) Un (ou des) carré(s) est pri
s lorsqu'il est
1280 REM compris entre deux carrés appar
tenant au joueur,
1290 REM que ce soit en ligne verticale,
 horizontale ou diagonale.
1300 REM
1310 REM  4) Un mouvement "illégal" est
signalé après 30 secondes
1320 REM par un signal sonore.
1330 REM 5) S'il n'y a plus de possibili
té de jouer en prise,
1340 REM il est alors permis de jouer n'
importe où.
1350 REM 6) Le gagnant est le joueur qui
 a le plus grand nombre
1360 REM de carrés dans sa couleur à la
fin du jeu.
1370 REM
1380 REM Ci-dessous une séquence de jeu
tout à fait valide
1390 REM lorsque l'on joue à deux
1400 REM et qui illustre une partie corr
ecte.
1410 REM MOUVEMENTS : 4-4,4-5,5-5,5-4,6-
4,6-5,4-6,3-5,6-6,6-3,
1420 REM 2-5,6-7,7-2,1-5,6-8,8-1,4-3,ETC
.
1430 CALL SCREEN(14)
1440 DIM EDGE(8),PLAY(2,10),EX(20),EY(20
),DIR(3),A(9,9)
1450 CALL CLEAR
1460 INPUT "INSTRUCTIONS (FAITES 1)":I
1470 IF I<>1 THEN 1510
1480 CALL CLEAR
1490 PRINT "POUR VOIR LES INSTRUCTIONS,
LISTEZ LES LIGNES 1000 A     1420 DU PROG
RAMME."
1500 STOP
1510 DATA 32,32,84,73,45,57,57,32
1520 RESTORE
1530 COMPLAY=0
1540 CALL CLEAR
1550 FOR I=1 TO 8
1560 READ PLAY(1,I)
1570 PLAY(2,I)=PLAY(1,I)
1580 NEXT I
1590 PRINT "UN JOUEUR SEUL     : 1
OU DEUX JOUEURS    : 2"
1600 INPUT COMP
1610 PRINT
1620 IF COMP=2 THEN 1820
1630 PRINT "FACILE     : 1
MOYEN    : 2                DIFFICILE :
3"
1640 INPUT "VOTRE CHOIX : ":HARD
1650 PRINT
1660 PRINT "TI-99 JOUE LE 1er  : 1
TI-99 JOUE LE 2me : 2"
1670 INPUT "VOTRE CHOIX : ":COMPLAY
1680 PRINT
1690 IF COMPLAY=1 THEN 1760
1700 PRINT "NOM DU 1er JOUEUR : "
1710 INPUT Z1$
1720 FOR I=1 TO 10
1730 CALL GCHAR(23,I+4,PLAY(1,I))
1740 NEXT I
1750 GO TO 1920
1760 PRINT "NOM DU 2eme JOUEUR"
1770 INPUT Z1$
1780 FOR I=1 TO 10
1790 CALL GCHAR(23,I+4,PLAY(2,I))
1800 NEXT I
1810 GO TO 1920
```

```
1820 PRINT "NOM DU 1er JOUEUR"
1830 INPUT Z1$
1840 FOR I=1 TO 10
1850 CALL GCHAR(23,I+4,PLAY(1,I))
1860 NEXT I
1870 PRINT "NOM DU 2eme JOUEUR"
1880 INPUT Z1$
1890 FOR I=1 TO 10
1900 CALL GCHAR(23,I+4,PLAY(2,I))
1910 NEXT I
1920 FOR I=1 TO 3
1930 DIR(I)=I-2
1940 NEXT I
1950 FOR I=0 TO 9
1960 FOR J=0 TO 9
1970 A(I,J)=1
1980 IF I*J*(J-9)*(I-9)<>0 THEN 2000
1990 A(I,J)=A(I,J)+1
2000 NEXT J
2010 NEXT I
2020 CALL CLEAR
2030 PRINT "            JOUE :    X=    Y="
2040 E$="FFFFFFFFFFFFFFFF"
2050 CALL CHAR(120,E$)
2060 CALL COLOR(12,2,8)
2070 FOR I=1 TO 8
2080 Y=3+2*I
2090 X=Y+4
2100 CD=48+I
2110 CALL VCHAR(Y,8,CD)
2120 CALL VCHAR(4,X,CD)
2130 NEXT I
2140 A$="FF80808080808080"
2150 B$="FF01010101010101"
2160 C$="80808080808080FF"
2170 D$="01010101010101FF"
2180 CALL CHAR(96,A$)
2190 CALL CHAR(97,B$)
2200 CALL CHAR(98,C$)
2210 CALL CHAR(99,D$)
2220 CALL CHAR(104,A$)
2230 CALL CHAR(105,B$)
2240 CALL CHAR(106,C$)
2250 CALL CHAR(107,D$)
2260 CALL CHAR(112,A$)
2270 CALL CHAR(113,B$)
2280 CALL CHAR(114,C$)
2290 CALL CHAR(115,D$)
2300 CALL COLOR(9,2,8)
2310 CALL COLOR(10,2,9)
2320 CALL COLOR(11,2,11)
2330 TYPE=1
2340 FOR X=1 TO 8
2350 FOR Y=1 TO 8
2360 GOSUB 3470
2370 NEXT Y
2380 NEXT X
2390 PL=1
2400 TURN=1
2410 REM ====
2420 REM DEBUT DU JEU
2430 FOR I=1 TO 10
2440 CALL HCHAR(23,I+1,PLAY(PL,I))
2450 NEXT I
2460 CALL VCHAR(23,23,120)
2470 CALL VCHAR(23,28,120)
2480 IF COMPLAY=PL THEN 2580
2490 CALL KEY(0,RE,ST)
2500 IF ST<>1 THEN 2490
2510 X=RE-48
2520 CALL VCHAR(23,23,RE)
2530 CALL KEY(0,RE,ST)
2540 IF ST<>1 THEN 2530
2550 Y=RE-48
2560 CALL VCHAR(23,28,RE)
2570 GO TO 2610
2580 GOSUB 4190
2590 CALL VCHAR(23,23,48+X)
2600 CALL VCHAR(23,28,48+Y)
2610 IF X>8 THEN 2760
2620 IF X<1 THEN 2760
2630 IF Y>8 THEN 2760
2640 IF Y<1 THEN 2760
2650 IF A(X,Y)>1 THEN 2760
2660 IF TURN<=4 THEN 2710
2670 GOSUB 3740
2680 IF EXTRA>0 THEN 2780
2690 GOSUB 3310
2700 IF SW=1 THEN 2760 ELSE 2780
2710 IF X>5 THEN 2760
2720 IF X<4 THEN 2760
2730 IF Y>5 THEN 2760
2740 IF Y<4 THEN 2760
2750 GOTO 2780
2760 CALL SOUND(500,200,3)
2770 GOTO 2460
2780 A(X,Y)=PL+2
2790 FOR I=X-1 TO X+1
2800 FOR J=Y-1 TO Y+1
2810 IF A(I,J)<>1 THEN 2830
2820 A(I,J)=0
2830 NEXT J
2840 NEXT I
2850 TURN=TURN+1
2860 TYPE=2
2870 IF PL=1 THEN 2890
2880 TYPE=3
2890 GOSUB 3470
2900 GOSUB 3740
2910 GOSUB 4060
2920 IF TURN=65 THEN 3000
2930 IF PL=1 THEN 2960
2940 PL=1
2950 GOTO 2970
2960 PL=2
2970 GOTO 2430
2980 REM
2990 REM FIN DU JEU ET CALCUL DES TOTAUX
```

```
3000 TOT1=0
3010 TOT2=0
3020 FOR I=1 TO 8
3030 FOR J=1 TO 8
3040 IF A(I,J)=3 THEN 3070
3050 TOT2=TOT2+1
3060 GO TO 3080
3070 TOT1=TOT1+1
3080 NEXT J
3090 NEXT I
3100 IF TOT2<>TOT1 THEN 3130
3110 PRINT "EGALITE 32 TO 32"
3120 GO TO 3230
3130 PRINT
3140 PRINT "AU SCORE DE";TOT1;" A";TOT2;
":"
3150 PRINT "          GAGNE"
3160 IF TOT1>TOT2 THEN 3190
3170 PWIN=2
3180 GO TO 3200
3190 PWIN=1
3200 FOR I=1 TO 10
3210 CALL HCHAR(23,I+1,PLAY(PWIN,I))
3220 NEXT I
3230 PRINT "POUR REJOUER : 1
POUR ARRETER : 2"
3240 INPUT GAME
3250 CALL CLEAR
3260 IF GAME=1 THEN 1520
3270 STOP
3280 REM FIN DU JEU
3290 REM
3300 REM LE JEU EST-IL LEGAL?
3310 SW=0
3320 ZZX=X
3330 ZZY=Y
3340 FOR X=1 TO 8
3350 FOR Y=1 TO 8
3360 IF A(X,Y)<>0 THEN 3400
3370 GOSUB 3740
3380 IF EXTRA=0 THEN 3400
3390 SW=1
3400 NEXT Y
3410 NEXT X
3420 X=ZZX
3430 Y=ZZY
3440 RETURN
3450 REM
3460 REM ROUTINE DES COULEURS
3470 IF TYPE<>1 THEN 3530
3480 S1=96
3490 S2=97
3500 S3=98
3510 S4=99
3520 GO TO 3630
3530 IF TYPE<>2 THEN 3590
3540 S1=104
3550 S2=105
3560 S3=106
3570 S4=107
3580 GO TO 3630
3590 S1=112
3600 S2=113
3610 S3=114
3620 S4=115
3630 X1=7+2*X
3640 X2=X1+1
3650 Y1=3+2*Y
3660 Y2=Y1+1
3670 CALL HCHAR(Y1,X1,S1)
3680 CALL HCHAR(Y1,X2,S2)
3690 CALL HCHAR(Y2,X1,S3)
3700 CALL HCHAR(Y2,X2,S4)
3710 RETURN
3720 REM
3730 REM  CARRES GAGNES
3740 EXTRA=1
3750 FOR I=1 TO 3
3760 FOR J=1 TO 3
3770 U=X+DIR(I)
3780 V=Y+DIR(J)
3790 IF U=X THEN 3800 ELSE 3810
3800 IF V=Y THEN 4000
3810 IF PL=1 THEN 3830
3820 IF A(U,V)=3 THEN 3840 ELSE 4000
3830 IF A(U,V)=4 THEN 3840 ELSE 4000
3840 U=U+DIR(I)
3850 V=V+DIR(J)
3860 IF A(U,V)<=1 THEN 4000
3870 IF A(U,V)=2 THEN 4000
3880 IF PL=1 THEN 3900
3890 IF A(U,V)=4 THEN 3910 ELSE 3840
3900 IF A(U,V)=3 THEN 3910 ELSE 3840
3910 U=X+DIR(I)
3920 V=Y+DIR(J)
3930 IF A(U,V)=PL+2 THEN 4000
3940 EX(EXTRA)=U
3950 EY(EXTRA)=V
3960 EXTRA=EXTRA+1
3970 U=U+DIR(I)
3980 V=V+DIR(J)
3990 GO TO 3930
4000 NEXT J
4010 NEXT I
4020 EXTRA=EXTRA-1
4030 RETURN
4040 REM
4050 REM  ADDITION DES CARRES DE COULEUR
S
4060 XX=X
4070 YY=Y
4080 FOR K=1 TO EXTRA
4090 X=EX(K)
4100 Y=EY(K)
4110 GOSUB 3470
4120 A(X,Y)=PL+2
4130 NEXT K
4140 X=XX
```

```
4150 Y=YY
4160 RETURN
4170 REM
4180 REM STRATEGIE DU TI-99
4190 IF TURN>4 THEN 4280
4200 FOR X=4 TO 5
4210 FOR Y=4 TO 5
4220 IF A(X,Y)>1 THEN 4250
4230 ZX=X
4240 ZY=Y
4250 NEXT Y
4260 NEXT X
4270 GO TO 5530
4280 EXTRAP=-100
4290 FOR X=1 TO 8
4300 FOR Y=1 TO 8
4310 IF A(X,Y)<>0 THEN 5430
4320 GOSUB 3740
4330 IF EXTRA=0 THEN 5430
4340 IF TURN<57 THEN 4360
4350 HARD=1
4360 IF HARD=1 THEN 5390
4370 IF X=1 THEN 4700
4380 IF X=8 THEN 4700
4390 IF Y=1 THEN 4580
4400 IF Y=8 THEN 4580
4410 IF HARD=2 THEN 5390
4420 IF X=2 THEN 4500
4430 IF X=7 THEN 4500
4440 IF Y=2 THEN 5390
4450 IF Y=7 THEN 5390
4460 IF X*X+Y*Y+36<>9*(X+Y)THEN 4480
4470 EXTRA=EXTRA+1.1
4480 EXTRA=EXTRA+4
4490 GO TO 5390
4500 IF Y=2 THEN 4530
4510 IF Y=7 THEN 4530
4520 GO TO 5390
4530 U=(7*X-9)/5
4540 V=(7*Y-9)/5
4550 IF A(U,V)>1 THEN 5390
4560 EXTRA=EXTRA-12
4570 GO TO 5390
4580 IF HARD=2 THEN 4680
4590 FOR I=1 TO 8
4600 EDGE(I)=A(I,Y)
4610 NEXT I
4620 FOR I=1 TO EXTRA
4630 IF EY(I)<>Y THEN 4650
4640 EDGE(EX(I))=PL+2
4650 NEXT I
4660 Z=X
4670 GO TO 4890
4680 EXTRA=EXTRA+3
4690 GO TO 5390
4700 IF Y=1 THEN 4840
4710 IF Y=8 THEN 4840
4720 IF HARD=2 THEN 4820
4730 FOR I=1 TO 8
4740 EDGE(I)=A(X,I)
4750 NEXT I
4760 FOR I=1 TO EXTRA
4770 IF EX(I)<>X THEN 4790
4780 EDGE(EY(I))=PL+2
4790 NEXT I
4800 Z=Y
4810 GO TO 4890
4820 EXTRA=EXTRA+3
4830 GO TO 5390
4840 IF HARD=2 THEN 4870
4850 EXTRA=EXTRA+14
4860 GO TO 5390
4870 EXTRA=EXTRA+6
4880 GO TO 5390
4890 CORN=0
4900 FOR I=1 TO EXTRA
4910 IF 2*(EX(I)-2)*(EX(I)-7)+(EY(I)-2)*
(EY(I)-7)=0 THEN 4930
4920 GO TO 4980
4930 U=(7*EX(I)-9)/5
4940 V=(7*EY(I)-9)/5
4950 IF A(U,V)<=1 THEN 4970
4960 GO TO 4980
4970 CORN=1
4980 NEXT I
4990 TYPE=0
5000 IF EDGE(1)=PL+2 THEN 5050
5010 LT=EDGE(1)
5020 IF LT<>0 THEN 5060
5030 LT=1
5040 GO TO 5060
5050 LT=2
5060 FOR I=2 TO Z-1
5070 IF EDGE(I)=PL+2 THEN 5120
5080 IF EDGE(I)<=1 THEN 5110
5090 LT=EDGE(I)
5100 GO TO 5120
5110 LT=1
5120 NEXT I
5130 IF LT=2 THEN 5290
5140 IF EDGE(8)=PL+2 THEN 5190
5150 HT=EDGE(8)
5160 IF HT<>0 THEN 5200
5170 HT=1
5180 GO TO 5200
5190 HT=2
5200 FOR I=7 TO Z+1 STEP -1
5210 IF EDGE(I)=PL+2 THEN 5260
5220 IF EDGE(I)<=1 THEN 5250
5230 HT=EDGE(I)
5240 GO TO 5260
5250 HT=1
5260 NEXT I
5270 IF HT=2 THEN 5290
5280 IF HT<>LT THEN 5350
5290 TYPE=1
5300 IF CORN=0 THEN 5330
5310 EXTRA=EXTRA-8
```

```
5320 GO TO 5390
5330 EXTRA=EXTRA+8
5340 GO TO 5390
5350 IF CORN=0 THEN 5380
5360 EXTRA=EXTRA-12
5370 GO TO 5390
5380 EXTRA=EXTRA-4
5390 IF EXTRA>EXTRAP THEN 5400 ELSE 5430
5400 EXTRAP=EXTRA
5410 ZX=X
5420 ZY=Y
5430 NEXT Y
5440 NEXT X
5450 IF EXTRAP=-100 THEN 5460 ELSE 5530
5460 FOR X=1 TO 8
5470 FOR Y=1 TO 8
5480 IF A(X,Y)>1 THEN 5510
5490 ZX=X
5500 ZY=Y
5510 NEXT Y
5520 NEXT X
5530 X=ZX
5540 Y=ZY
5550 RETURN
```

Accélérez vos programmes en Basic Etendu

Si certains programmes que vous avez écrits en Basic Etendu n'utilisent pas de sprites, mettez en tête la ligne suivante :

10 CALL LOAD(-31878,0)

L'adresse 31878 est en effet celle où se fait le guidage de tous les sprites. Si votre programme n'en utilise pas, cette ligne supprime cette procédure et accélère d'autant les autres.

Si vous utilisez par exemple cinq sprites, alors remplacez le zéro par un cinq. Vous pouvez donc dès le départ du programme décider du nombre de sprites à gérer. Moins il y en aura, plus rapide sera le déroulement du reste de l'action. Vous ne pouvez évidemment pas mettre dans cette ligne un nombre supérieur à 28, nombre de sprites maximal du Basic Etendu.

Astéroïdes

Edouard Jouan

Ce programme que nous devons à nos amis du club TISOFT-HOME de Bruxelles est écrit en Basic. Il nécessite donc ce module pour pouvoir fonctionner. Il peut en outre utiliser le Speech Synthetizer. Si vous possédez cette unité, le déroulement du jeu en sera beaucoup plus spectaculaire.

Le jeu

Le principe du jeu en lui-même est simple : un vaisseau spatial se déplace au milieu d'un nuage d'astéroïdes, à une vitesse de plus en plus grande. Il faut bien entendu tenir le plus longtemps possible en évitant toute collision avec l'un de ces objets. A ce sujet, avez-vous entendu, un jour à la radio, un docte savant déclarer : "Non, cela n'était pas un O.V.N.I.! C'était seulement un objet volant que l'on n'a pas pu identifier..." ?

Les asteroïdes sont symbolisés par les sprites 2 à 28; leur forme est celle du caractère 97 (ligne 370). Le vaisseau spatial, quant à lui, est représenté par le sprite 1 sous la forme du caractère 96 (ligne 390). La ligne 410 détermine la vitesse à laquelle se déplacent les obstacles, les lignes 480 et 500 font déplacer le vaisseau vers la gauche ou vers la droite...

Commandes spéciales

Bref, la lecture du programme semble assez évidente. Je vous signale cependant particulièrement les lignes concernant le Speech Synthetizer :

- Remarquez, notamment à la ligne 130, la présence du signe " + " entre les mots. Cela donne à l'ordinateur un ton plus "coulé" en liant les mots les uns aux autres. Si l'on met simplement un blanc, le ton est trop haché et cela a vraiment l'air de venir d'un ordinateur...

- La ligne 710 consiste à faire dire par le TI un message d'alerte, puis à vous encourager à rejouer. Les dièses autour de "TRY AGAIN" servent à bien préciser qu'il s'agit là d'une expression faisant partie en tant que telle du dictionnaire du Speech Synthetizer. L'expression est alors prononcée sur un ton d'encouragement, comme s'il s'agissait d'un seul mot. Sans cette indication, le TI-99 dirait simplement les deux mots "TRY" puis "AGAIN", ce qui est totalement différent. Il en va de même pour toutes les autres expressions du dictionnaire comme par exemple l'expression "TEXAS INTRUMENTS" qui doit être écrite "TEXAS INSTRUMENTS" : sans cela, l'ordinateur ne fait qu'épeler ces deux mots qu'il ne connaît pas sous une autre forme.

Remarquez aussi la ligne 530 : CALL COINC(ALL,A). Son objet est de tester la coïncidence entre deux quelconques sprites. Tant qu'il n'y a pas de coïncidence, la valeur de A est nulle, elle devient égale à -1 dans le cas contraire. Tous les asteroïdes se déplaçant dans le même sens (de haut en bas), seul le vaisseau peut les rencontrer, ce qui envoie le programme vers la procédure de collision (ligne 540).

Il faut noter que l'instruction CALL COINC reste valable même si les sprites ne sont plus présents à l'écran : entre le moment où ils disparaissent d'un côté et réapparaissent de l'autre, il y a toujours un petit délai. Si une collision se produit pendant ce délai, vous ne la voyez pas mais elle vous est quand-même signalée.

Augmenter la difficulté

On peut aisément augmenter la difficulté de ce jeu en grossissant les formes des objets. Il faut alors inclure au début du programme une instruction CALL MAGNIFY(X) :

- X = 1 : aucun changement;

- X = 2 : la dimension de tous les sprites est doublée ;

- X = 3 : les sprites sont alors dessi-

nés dans une grille de 16 sur 16 (au lieu de 8 sur 8). Il convient de redéfinir leur forme dans les lignes 330 et 340.

- X = 4 : Les sprites sont dessinés en 16 sur 16 et leur taille est doublée, ce qui revient à dire qu'on les mutiplie par quatre.

La dimension des astéroïdes et celle du vaisseau étant augmentées, il reste moins d'espace pour circuler et le jeu est plus difficile.

Astéroïdes - Basic Etendu

```
50 REM ============================
60 REM COPYRIGHT 99 MAGAZINE
70 REM     ET TISOFT-HOME.
80 REM ============================
90 REM
100 CALL CLEAR
110 INPUT "Le SPEECH SYNTHESIZER est-il
branche (O/N) ?":SS$
120 IF SS$="N" THEN 250
130 CALL SAY("THIS+IS+THE1+PROGRAM+OF")
140 DISPLAY AT(12,12):"ASTEROID"
150 FOR O=1 TO 5
160 FOR I=550 TO 555
170 CALL SOUND(-100,I,1)
180 NEXT I
190 FOR I=554 TO 549
200 CALL SOUND(-100,I,1)
210 NEXT I
220 NEXT O
230 CALL SAY("YOU+HAVE+JUST+TO+PRESS+RIG
HT+OR+LEFT+KEY+TO+MOVE+RIGHT+OR+LEFT")
240 CALL SAY("GOOD+GAMES")
250 REM ASTEROIDS
260 CALL CLEAR
270 RANDOMIZE
280 CALL SCREEN(2)
290 VIT=1
300 FOR I=2 TO 8
310 CALL COLOR(I,16,5)
320 NEXT I
330 CALL CHAR(96,"101038BAFEFEC682")
340 CALL CHAR(97,"02373F7F7F7F7A58")
350 FOR I=28 TO 2 STEP -1
360 CALL SOUND(-3000,I*100+110,10)
370 CALL SPRITE(#I,97,13*RND+3,1,I*9,126
*RND+1,0)
380 NEXT I
390 CALL SPRITE(#1,96,5,180,125)
400 FOR I=2 TO 28
410 CALL MOTION(#I,VIT,0)
420 NEXT I
430 FOR I=2 TO 28
440 CALL KEY(0,KEY,STATUS)
450 IF KEY=83 THEN 480
460 IF KEY=68 THEN 500
470 GOTO 510
480 CALL MOTION(#1,0,-10)
490 GOTO 510
500 CALL MOTION(#1,0,10)
510 CALL MOTION(#I,VIT,0)
520 CALL SOUND(-1000,-5,20,VIT*20+110,15
)
530 CALL COINC(ALL,A)
540 IF A=-1 THEN 630
550 NEXT I
560 VIT=VIT+1
570 IF VIT>=30 THEN 760
580 POINTS=POINTS+INT(VIT/2)
590 DISPLAY AT(2,2):"POINTS ";POINTS
600 DISPLAY AT(3,2):"VITESSE";VIT
610 DISPLAY AT(4,2):"COLLISIONS";COLLISI
ON
620 GOTO 430
630 POINTS=POINTS-VIT*2
640 CALL MOTION(#1,0,0)
650 FOR I=1 TO 5
660 CALL SCREEN(7)
670 CALL SOUND(-500,-7,1)
680 CALL SCREEN(2)
690 NEXT I
700 IF SS$="N" THEN 720
710 CALL SAY("S O S,S O S,,#TRY AGAIN#")
720 COLLISION=COLLISION+1
730 IF COLLISION=10 THEN 960
740 VIT=1
750 GOTO 430
760 FOR I=1 TO 28
770 CALL MOTION(#I,0,0)
780 NEXT I
790 P$="
EN FIN DE COURSE VOUS OBTENEZ "
800 A$=STR$(POINTS)
810 R$=" POINTS ET AVEZ PROVOQUE "
820 C$=STR$(COLLISION)
830 O$=" COLLISIONS.....................
.........."
840 M$=P$&A$&R$&C$&O$
850 FOR I=1 TO LEN(M$)
860 DISPLAY AT(12,1):SEG$(M$,I,28)
870 CALL SOUND(1,28*100+110,10)
880 NEXT I
890 CALL CLEAR
900 INPUT "DESIREZ-VOUS REJOUER (O/N)?":
M$
910 IF M$="N" THEN END
920 VIT=10
930 COLLISION=0
940 POINTS=0
950 GOTO 250
960 IF SS$="N" THEN 890
970 CALL SAY("OH,,YOU+ARE+OUT+OF+COURSE,
,,SORRY+THE1+GAMES+IS+OVER")
980 GOTO 890
```

Kermit la grenouille

Alexandre Duback

Kermit est une charmante petite grenouille verte à laquelle, vous vous en doutez, il arrive des tas d'ennuis. Elle se trouve en effet au bord d'une route sur laquelle passent des camions en très grand nombre. Bien entendu, elle n'a qu'une envie, celle de traverser cette route.

Une fois franchi ce dangereux obstacle, elle trouve une rivière. Pas de problème, direz-vous, pour une grenouille. Eh bien si ! Kermit, encore un peu jeune sans doute, ne sait pas nager. Elle doit donc attendre que des feuilles flottant sur les flots bleus passent à sa proximité pour sauter dessus et, progressant ainsi de feuille en feuille, atteindre l'autre rive. Mais elle est étrange cette rivière : certaines feuilles vont dans un sens, que l'on peut imaginer comme celui du courant, mais d'autres vont en sens contraire. Influence du vent sans doute...

La rivière étant traversée, il reste encore une route, avec toujours autant de ces énormes monstres que sont les camions lancés par ces dingues d'humains à des vitesses folles. Et pour eux, la vie d'une grenouille... Pensez donc ! Il vaut mieux essayer de se faufiler que de compter qu'ils vont faire attention.

Lorsque Kermit a terminé son trajet, elle se retrouve à son point de départ, avec des obstacles toujours pareils. Et si elle n'arrive pas, c'est l'une de ses soeurs qui prendra sa place (Kermit fait partie d'une famille de cinq enfants, vous ne saviez pas ?). Le score du joueur est le nombre de traversées réussies par les cinq petites grenouilles.

Vous avez certainement remarqué que ce programme est en Basic Etendu. Il utilise abondamment les sprites (ou lutins) et je crois qu'une petite révision des instructions les concernant ne sera pas inutile.

CALL SPRITE

CALL SPRITE (-numéro de sprite, numéro de caractère, couleur, ligne, colonne, vitesse horizontale, vitesse verticale)

Vous pouvez utiliser 28 sprites, numérotés de 1 à 28. Leur forme est, préalablement à l'utilisation de cette instruction, définie par un CALL CHAR. Le numéro de ce caractère est donc celui qui est inscrit ici. La couleur est définie pour tout le sprite, le fond étant obligatoirement transparent. Les autres valeurs ne posent pas de problème.

Sachez que lorsque deux sprites se rencontrent, celui qui est de numéro le plus faible passe devant l'autre, c'est-à-dire que, par exemple, le sprite -1 peut masquer le sprite -2 mais jamais il ne sera masqué par lui.

CALL COINC

CALL COINC (-numéro, -numéro, tolérance, variable numérique)

A la place du second numéro, on peut mettre des numéros de ligne et de colonne (non précédés du signe -). On peut encore écrire :

CALL COINC (ALL, variable numérique)

Cette instruction teste la rencontre entre deux ou plusieurs sprites. Si une telle rencontre se produit, la valeur 1 est attribuée à la variable numérique. Dans le cas contraire, cette variable est égale à zéro.

La notion de tolérance est importante : elle détermine en fait le niveau de difficulté du jeu. Une tolérance importante permet en effet de ne faire se chevaucher les sprites qu'approximativement, sinon pas du tout. A la limite, il suffit que le premier sprite soit proche du deuxième pour que la valeur de la variable soit mise à 1.

Si vous voulez "corser" ce jeu, après avoir atteint le niveau d'un vrai professionnel, il vous suffit donc de modifier en conséquence toutes les instructions CALL SPRITE.

CALL MOTION

CALL MOTION (-numéro, vitesse horizontale, vitesse verticale)

Cette instruction se contente de modifier les caractéristiques de déplacement d'un sprite. Dans ce programme, elle est essentielle : Kermit avance lorsque l'on appuie sur E, stoppe lorsque l'on relève le doigt, se déplace horizontalement vers la gauche ou la droite selon la feuille sur laquelle elle se trouve, etc. Toutes ces modifications de trajet sont obtenues par CALL MOTION.

Il ne faut surtout pas oublier que lorsqu'un sprite est lancé, il ne stoppe ou ne modifie sa trajectoire que si on lui en donne expressément l'instruction. Essayez le petit programme suivant :

```
10 CALL CHAR(100,"F")
20 CALL SPRITE(1,100,2,10,12,5,5)
30 GOTO 30
```

Tant que vous n'aurez pas fait "CLEAR", vous verrez le sprite faire une trajectoire constante. Modifiez maintenant :

```
30 FOR DELAI = 1 TO 500
40 NEXT DELAI
50 CALL MOTION(-1,0,0)
60 GOTO 60
```

Vous voyez alors le sprite faire un petit bout de trajet (le temps que la boucle vide se termine) puis stopper dans un coin de l'écran, lorsque le programme exécute la ligne 50.

NB : la ligne 60 a pour objet de faire tourner le programme sur lui-même dans l'attente d'un "CLEAR". Il faut utiliser ce truc, sans quoi le programme s'arrête et le sprite disparait.

CALL PATTERN

CALL PATTERN (-numéro, numéro de caractère)

Cette instruction modifie la forme d'un sprite en lui attribuant une forme définie par un autre caractère que celle primitivement attribué dans CALL SPRITE. Il en est ainsi, par exemple, lorsque Kermit se fait écraser par un camion : elle devient une horrible tache verte sur la route (que les âmes sensibles évitent donc ce jeu !).

Conclusion

Mes enfants utilisent beaucoup ce programme qu'ils trouvent très "sympa". J'ai été obligé, pour pouvoir comparer mes scores aux leurs, de leur fournir une version plus difficile (avec une plus petite tolérance). Et ces petits diables continuent de progresser...

```
50 REM *****************************
60 REM *   KERMIT LA GRENOUILLE    *
70 REM *  COPYRIGHT 99 MAGAZINE    *
80 REM *      ET J. DE WINTER      *
90 REM =============================
100 REM PREPARATION DE L'ECRAN
110 REM =============================
120 CALL CLEAR :: CALL SCREEN(2):: CALL
COLOR(14,8,8):: CALL COLOR(13,6,6):: CAL
L COLOR(12,15,15)
130 CALL COLOR(2,16,15):: CALL DELSPRITE
(ALL):: CALL MAGNIFY(3)
140 CALL COLOR(1,6,16):: FOR I=5 TO 8 ::
 CALL COLOR(I,6,16):: NEXT I
150 CALL COLOR(9,10,10)
160 REM =============================
170 REM CONSTRUCTION DES CARACTERES
180 REM =============================
190 CALL CHAR(33,"FFF8E0C0C0808080")
200 CALL CHAR(34,"FF1F070303010101")
210 CALL CHAR(35,"0101010303071FFF")
220 CALL CHAR(36,"00071F3F3F7FFFFF")
230 CALL CHAR(37,"000000FFFFFFFFFF")
240 CALL CHAR(38,"FFFFFFFFFF")
250 CALL CHAR(40,"0000FFFFFFFF")
260 CALL CHAR(136,"00000000FFFFFFFFFFFF
FFF0000000000000000CEFEFFFFFFFFFFECE")
270 CALL CHAR(132,"00000000737FFFFFFFFF7
F730000000000000000FFFFFFFFFFFFFFFF")
280 CALL CHAR(128,"0107077F7F7F7FFFFFFFF
F7F1F0F0F0388FCFCFCFCFCFEFFFFFFFFFEFCF8F
0A0")
290 CALL CHAR(124,"070F0F1F7F7FFFFFFF7FF
F7F3F1F0F0780F8F8F8F8FEFEFFFFF8F8F8FFF8C
080")
300 CALL CHAR(120,"01037FFF7F7F7FFFFF7F7
F1F1F1F1F01E0F0F8FCFEFFFEFFFFFFFEFEFCF0F
0E0")
310 CALL CHAR(116,"050F1F3F7FFFFFFFFFF7F3
F3F3F3F3F11C0F0F0F8FEFFFFFFFFFFEFEFEFEE0E
080")
320 CALL CHAR(112,"010307031B070F2F2F2F3
F1B0000000080C0E0C0D8E0F0F4F4F4FCD8")
330 CALL CHAR(108,"02070F0F0FBFBFFFFFFFF
FFF7D38100080E0E0F0FCFFFFFFFFFFEFCF0E0400
0")
340 CALL CHAR(104,"0F30404788939797979 79
3884740300FF00C02E211C9E9E9E9E9C911E2020
CF0")
350 CALL CHAR(100,"000003040B14292B2B291
40B0403000000000C020D02894D4D49428D020C")
360 CALL CHAR(96,"00000000000102050502 01
000000000000000000008040A0A0408")
370 GOSUB 1780
380 CALL SPRITE(#9,112,13,140,65)
390 FOR I=10 TO 16 :: CALL HCHAR(I,10,32
,20):: NEXT I
400 DISPLAY AT(11,9)SIZE(19):"MON NOM ES
T KERMIT"
410 DISPLAY AT(13,9)SIZE(19):"FAITES 'S'
 POUR"
420 DISPLAY AT(15,9)SIZE(19):"COMMENCER
LE JEU."
430 CALL HCHAR(10,10,33):: CALL HCHAR(10
,29,34):: CALL HCHAR(16,29,35):: CALL HC
HAR(17,10,36)
440 X=X
450 X=X
460 GOSUB 1730
470 CALL DELSPRITE(#9):: CALL HCHAR(9,1,
130,288)
480 FOR I=3 TO 8 :: CALL COLOR(I,7,16)::
 NEXT I
490 REM =============================
500 REM  PLACEMENT DES SPRITES
510 REM =============================
520 OVER=0
530 CALL SPRITE(#1,136,5,168,1,0,36)
540 CALL SPRITE(#2,136,5,168,140,0,36)
550 CALL SPRITE(#3,132,5,154,256,0,-31)
560 CALL SPRITE(#4,132,5,154,130,0,-31)
570 CALL SPRITE(#5,136,5,19,56,0,32)
580 CALL SPRITE(#6,136,5,19,196,0,32)
590 CALL SPRITE(#7,132,5,33,200,0,-39)
600 CALL SPRITE(#8,132,5,33,60,0,-39)
610 CALL SPRITE(#10,128,11,70,40,0,8)
620 CALL SPRITE(#11,124,12,70,123,0,8)
630 CALL SPRITE(#12,120,12,70,206,0,8)
640 CALL SPRITE(#13,116,11,92,20,0,-7)
650 CALL SPRITE(#14,128,11,92,104,0,-7)
660 CALL SPRITE(#15,124,12,92,196,0,-7)
670 CALL SPRITE(#16,120,11,116,50,0,6)
680 CALL SPRITE(#17,116,12,116,136,0,6)
690 CALL SPRITE(#18,128,11,116,224,0,6)
700 KER=1
710 CALL SPRITE(#9,112,13,184,129,0,0)
720 REM =============================
730 REM  DEBUT DU JEU
740 REM =============================
750 PLAT=0
760 DISPLAY AT(1,20)SIZE(8):"KERMIT";KER
770 GOSUB 1730
780 CALL LOCATE(#9,168,130)
790 GOSUB 1760 :: S=1 :: S1=2
800 CALL KEY(1,L,LL)
810 GOSUB 1580
820 IF PLAT=1 THEN 1380
830 IF LL=0 THEN 800
840 CALL LOCATE(#9,154,130)
850 GOSUB 1760 :: S=3 :: S1=4
860 CALL KEY(1,L,LL)
870 GOSUB 1580
880 IF PLAT=1 THEN 1380
890 IF LL=0 THEN 860
900 CALL LOCATE(#9,140,130)
910 GOSUB 1760
920 GOSUB 1730
930 CALL LOCATE(#9,116,132)
```

```
940 CALL COINC(#9,#16,4,D1):: CALL COINC(#9,#17,4,D2):: CALL COINC(#9,#18,4,D3)
950 IF D1<>0 OR D2<>0 OR D3<>0 THEN CALL MOTION(#9,0,6):: GOSUB 1760 :: GOTO 970
960 Y=116 :: X=134 :: GOTO 1620
970 CALL KEY(1,L,LL):: CALL POSITION(#9,Y,X)
980 IF X>248 THEN 1380
990 IF LL=0 THEN 970
1000 CALL POSITION(#9,Y,X)
1010 CALL LOCATE(#9,92,X):: CALL COINC(#9,#13,4,D1):: CALL COINC(#9,#14,4,D2):: CALL COINC(#9,#15,4,D3)
1020 IF D1<>0 OR D2<>0 OR D3<>0 THEN CALL SPRITE(#9,112,13,92,X-0,0,-7):: GOTO 1040
1030 Y=92 :: GOTO 1620
1040 GOSUB 1760
1050 CALL KEY(1,L,LL):: CALL POSITION(#9,Y,X)
1060 IF X<5 THEN 1380
1070 IF LL=0 THEN 1050
1080 CALL POSITION(#9,Y,X)
1090 CALL LOCATE(#9,70,X):: CALL COINC(#9,#10,4,D1):: CALL COINC(#9,#11,4,D2):: CALL COINC(#9,#12,4,D3)
1100 IF D1<>0 OR D2<>0 OR D3<>0 THEN CALL SPRITE(#9,112,13,70,X+0,0,8):: GOTO 1120
1110 Y=70 :: GOTO 1620
1120 GOSUB 1760
1130 CALL KEY(1,L,LL):: CALL POSITION(#9,Y,X)
1140 IF X>248 THEN 1380
1150 IF LL=0 THEN 1130
1160 GOSUB 1760
1170 CALL POSITION(#9,Y,X)
1180 CALL LOCATE(#9,50,X):: CALL MOTION(#9,0,0)
1190 IF X<118 OR X>138 THEN Y=65 :: GOSUB 1620 :: GOTO 1380
1200 GOSUB 1730
1210 GOSUB 1760 :: CALL LOCATE(#9,33,X):: S=7 :: S1=8
1220 CALL KEY(1,L,LL)
1230 GOSUB 1580
1240 IF PLAT=1 THEN 1380
1250 IF LL=0 THEN 1220
1260 GOSUB 1750
1270 CALL LOCATE(#9,19,X)
1280 S=5 :: S1=6
1290 CALL KEY(1,L,LL)
1300 GOSUB 1580
1310 IF PLAT=1 THEN 1380
1320 IF LL=0 THEN 1290
1330 GOSUB 1750 :: CALL LOCATE(#9,5,X)
1340 FOR DEL=1 TO 400 :: NEXT DEL
1350 J=J+1
1360 X=X
1370 OVER=OVER+1 :: GOTO 710
1380 KER=KER+1 :: IF KER<>6 THEN 710
1390 CALL HCHAR(1,1,140,32):: CALL DELSPRITE(ALL)
1400 CALL SPRITE(#9,112,13,140,46)
1410 FOR I=10 TO 16 :: CALL HCHAR(I,8,32,23):: NEXT I
1420 DISPLAY AT(11,7)SIZE(21):"TRAVERSEES :";OVER
1430 IF OVER=0 THEN M$="VOUS ETES NUL !" :: GOTO 1480
1440 IF OVER<5 THEN M$="MOYEN" :: GOTO 1480
1450 IF OVER<10 THEN M$="AMATEUR" :: GOTO 1480
1460 IF OVER<19 THEN M$="TRES BON !" :: GOTO 1480
1470 M$="PROFESSIONNEL"
1480 DISPLAY AT(13,7)SIZE(19):M$
1490 CALL HCHAR(10,8,33):: CALL HCHAR(10,30,34):: CALL HCHAR(16,30,35):: CALL HCHAR(17,8,36)
1500 DISPLAY AT(15,7)SIZE(19):"ENCORE UN COUP OçN?"
1510 CALL KEY(0,ST,ED):: CALL KEY(1,TS,DE)
1520 IF ST=78 THEN 370
1530 IF ST<>79 AND TS<>23 THEN 1510
1540 CALL DELSPRITE(#9):: CALL HCHAR(9,1,130,288):: GOTO 500
1550 REM ============================
1560 REM  TEST ROUTE
1570 REM ============================
1580 CALL COINC(#9,#S,20,TEST):: CALL COINC(#9,#S1,20,TEST1)
1590 IF TEST=0 AND TEST1=0 THEN RETURN ELSE CALL SOUND(600,-7,0)
1600 CALL PATTERN(#9,108):: PLAT=1 :: FOR DEL=1 TO 400 :: NEXT DEL
1610 RETURN
1620 REM   ACCIDENT
1630 CALL DELSPRITE(#9)
1640 CALL SPRITE(#20,104,16,Y,X)
1650 CALL SOUND(800,111,2)
1660 FOR DEL=1 TO 50 :: NEXT DEL
1670 CALL PATTERN(#20,100)
1680 FOR DEL=1 TO 100 :: NEXT DEL
1690 CALL PATTERN(#20,96)
1700 FOR DEL=1 TO 160 :: NEXT DEL
1710 CALL DELSPRITE(#20)
1720 GOTO 1380
1730 CALL KEY(1,L,LL)
1740 IF LL=0 THEN 1730 ELSE RETURN
1750 REM Ne supprimez pas cette ligne
1760 CALL SOUND(60,-3,0)
1770 RETURN
1780 CALL HCHAR(1,1,140,64)
1790 CALL HCHAR(3,1,120,128)
1800 CALL HCHAR(7,1,140,32)
```

```
1810 CALL HCHAR(8,1,37,32)
1820 CALL HCHAR(9,1,130,288)
1830 CALL HCHAR(18,1,38,32)
1840 CALL HCHAR(19,1,140,32)
1850 CALL HCHAR(20,1,120,128)
1860 CALL HCHAR(24,1,140,32)
1870 FOR TT=0 TO 17 STEP 17 :: FOR T=3 T
O 6 :: CALL HCHAR(T+TT,17,40,2):: NEXT T
 :: NEXT TT
1880 FOR TT=0 TO 11 STEP 11 :: FOR T=7 T
O 8 :: CALL HCHAR(T+TT,16,100,4):: NEXT
T :: NEXT TT
1890 RETURN
```

Le Biorythme

Thomas Coppens

Si vous habitez une grande ville, vous avez sans doute remarqué, dans certains endroits proches du centre commercial, des gens qui proposent aux passants leur horoscope et leur biorythme calculé au moyen d'un ordinateur. Mellab Karim s'est inspiré de cette idée et, faute de pouvoir faire un horoscope (trop littéraire pour les capacités du TI-99), a fait le programme du biorythme.

Je pense que vous savez ce dont il s'agit : une théorie, d'origine américaine, affirme que chacun de nous passe par des périodes de "haut" et de "bas" au cours desquelles il est plus ou moins en forme. Les inventeurs de cette théorie prétendent que ces périodes s'organisent en cycles réguliers ayant leur origine à la naissance de l'individu. Il est donc possible, en calculant l'évolution de ce cycle jusqu'à une date donnée, de déterminer avec précision le niveau auquel se trouve l'individu à cette date.

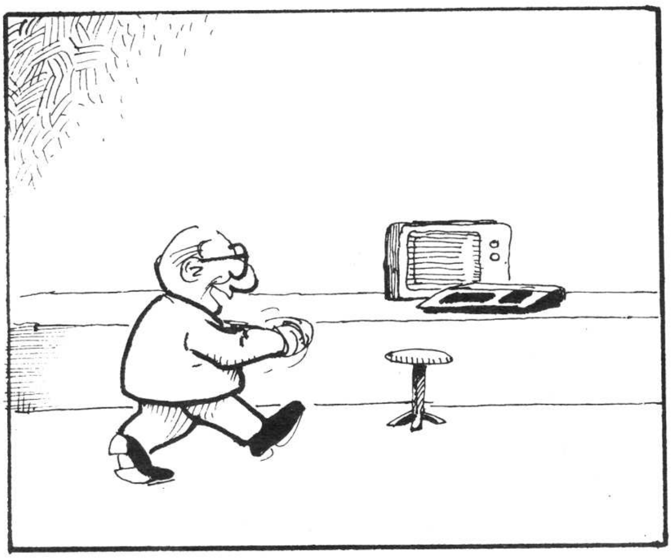
Les trois cycles du biorythme sont les suivants :

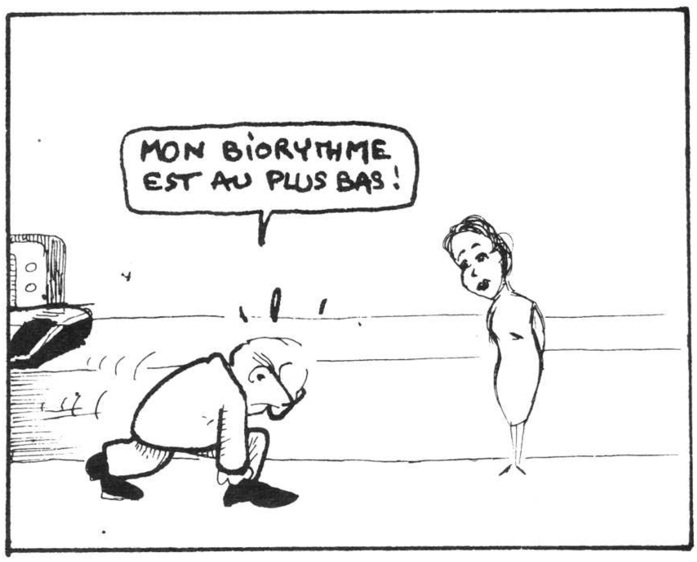
- forme physique : 23 jours;
- sentiments : 28 jours;
- intelligence : 33 jours.

Chacun de ces cycles se représente par une courbe sinusoïdale dont la période est égale à la durée du cycle, et dont l'origine se situe au point zéro au jour de la naissance.

Le programme que nous vous proposons réalise le calcul des trois courbes du biorythme et peut vous les présenter sous deux formes différentes : en graphique ou en texte indiquant les valeurs quotidiennes. De la sorte, si vous disposez d'une imprimante, vous pouvez, après les aménagements éventuellement nécessaires de ce programme, d'une part afficher le graphique et, d'autre part, éditer les textes. Remarquez qu'en outre le programme vous indique le jour de la semaine.

Lorsque vous taperez ce programme, vous remarquerez dans le listing certains signes bizarres : ne vous inquiétez pas, c'est mon imprimante qui est réglée pour les sortir. En fait, il s'agit simplement des crochets ouvrant et fermant. Tapez alors ces signes à la place de tout autre signe que vous ne trouvez pas sur votre clavier.






```
100 CALL SCREEN(5)
110 REM ****************************
120 REM * COPYRIGHT 99 MAGAZINE    *
130 REM *     ET MELLAB KARIM      *
140 REM ****************************
150 CALL CLEAR
160 FOR I=1 TO 3
170 READ G$(I)
180 NEXT I
190 DATA "PHYSIQUE","SENSIBILITE","INTEL
LECTUEL"
200 FOR I=1 TO 12
210 CALL COLOR(I,16,1)
220 NEXT I
```

```
230 CALL CHAR(138,"007844447840404")
240 CALL COLOR(14,5,16)
250 CALL CHAR(148,"0038444038044438")
260 CALL COLOR(13,2,1)
270 CALL COLOR(15,16,14)
280 CALL COLOR(16,11,7)
290 CALL CHAR(158,"0038101010101038")
300 FOR I=1 TO 7
310 READ C$(I)
320 NEXT I
330 DATA " VEN"," SAM"," DIM"," LUN"," M
AR"," MER"," JEU",
340 CALL CHAR(128,"0010387C381")
350 DEF PI=3.14159
360 GOSUB 2740
370 CALL CLEAR
380 B=23
390 V$="* BIORYTHME *"
400 H=2
410 VE=5
420 GOSUB 1830
430 V$="*************"
440 H=1
450 GOSUB 1830
460 H=3
470 GOSUB 1830
480 V$="BIORYTHME GRAPH OU NUM?"
490 H=5
500 VE=3
510 GOSUB 1830
520 V$="APPUYEZ SUR:1)GRAP"
530 H=6
540 GOSUB 1830
550 V$="2)NUM"
560 H=7
570 VE=15
580 GOSUB 1830
590 CALL SOUND(200,1800,6)
600 CALL KEY(0,KE,ST)
610 IF (ST=0)+(KE<49)+(KE>50)THEN 600
620 CALL SOUND(200,1100,2)
630 GH=KE-48
640 CALL CLEAR
650 INPUT "Voulez_vous donner votre date
 de naissance?°JJ,MM,AAAA§:":J,M,A
660 XA=A/4
670 YA=INT(XA)
680 ZA=XA-YA
690 GOSUB 2900
700 PRINT : :
710 GOSUB 1270
720 HH=N
730 INPUT "Donnez la date pour le bioryt
hme?°JJ,MM,AAAA§:":J,M,A
740 XA=A/4
750 YA=INT(XA)
760 ZA=XA-YA
770 GOSUB 2900
780 J1=J
790 M1=M
800 A1=A
810 A3=A1/4
820 A2=INT(A3*10)
830 A1=(A2/10)-INT(A3)
840 GOSUB 1270
850 GOTO 3000
860 P=N-HH
870 PRINT : :
880 JJ=N
890 INPUT "Pour combien de jour le biory
thme?:":BG
900 IF GH=2 THEN 2530
910 CALL CLEAR
920 V$=" DATE      °-§      °0§   °+§"
930 H=1
940 VE=0
950 GOSUB 1830
960 CALL CHAR(33,"000000FF")
970 CALL HCHAR(2,1,33,32)
980 CALL VCHAR(3,21,58,22)
990 ZZ=128
1000 VV=ZZ
1010 H=3
1020 V=P
1030 FOR BB=3 TO BG+2
1040 II=BB
1050 B=23
1060 ZZ=128
1070 IF BB>24 THEN 1080 ELSE 1150
1080 II=24
1090 CALL SOUND(300,1900,6)
1100 CALL KEY(0,KEY,STATUS)
1110 IF STATUS=0 THEN 1100
1120 PRINT
1130 CALL HCHAR(24,21,58)
1140 H=24
1150 FOR J=1 TO 3
1160 GOSUB 1340
1170 B=B+5
1180 ZZ=ZZ+10
1190 VV=ZZ
1200 GOSUB 1440
1210 NEXT J
1220 V=V+1
1230 JJ=JJ+1
1240 GOSUB 1510
1250 NEXT BB
1260 GOTO 2400
1270 IF M-3>0 THEN 1280 ELSE 1300
1280 M=M+1
1290 GOTO 1320
1300 A=A-1
1310 M=M+13
1320 N=INT(365.25*A)+INT(30.6*M)+J
1330 RETURN
1340 Z=V/B
1350 X=INT(Z)
1360 X=Z-X
```

```
1370 X=X*2
1380 T=X*PI
1390 R=SIN(T)
1400 E=R*100
1410 W=INT(E)
1420 W=W/100
1430 RETURN
1440 A=(W*9)+21
1450 CALL GCHAR(II,A,GG)
1460 IF (GG=32)+(GG=58)THEN 1480
1470 ZZ=128
1480 CALL HCHAR(II,A,ZZ)
1490 ZZ=VV
1500 RETURN
1510 P=(JJ-1)/7
1520 F=INT(P*10)
1530 L=ABS(INT(P)-(F/10))
1540 IF L=0 THEN 1610
1550 IF L=.1 THEN 1630
1560 IF L=.2 THEN 1650
1570 IF L=.4 THEN 1670
1580 IF L=.5 THEN 1690
1590 IF L=.7 THEN 1710
1600 IF L=.8 THEN 1730
1610 N$=C$(1)
1620 GOTO 1740
1630 N$=C$(2)
1640 GOTO 1740
1650 N$=C$(3)
1660 GOTO 1740
1670 N$=C$(4)
1680 GOTO 1740
1690 N$=C$(5)
1700 GOTO 1740
1710 N$=C$(6)
1720 GOTO 1740
1730 N$=C$(7)
1740 V$=N$&":"&STR$(J1)&"/"&STR$(M1)
1750 VE=0
1760 GOSUB 1830
1770 J1=J1+1
1780 IF J1>28 THEN 1790 ELSE 1810
1790 IF H>24 THEN 1820
1800 GOSUB 1880
1810 H=H+1
1820 RETURN
1830 FOR I=1 TO LEN(V$)
1840 AS=ASC(SEG$(V$,I,1))
1850 CALL HCHAR(H,VE+I,AS)
1860 NEXT I
1870 RETURN
1880 IF (J1>31)*(M1=1)THEN 1890 ELSE 192
0
1890 J1=1
1900 M1=2
1910 RETURN
1920 IF (J1>28)*(M1=2)*(A1<>0)THEN 1930
ELSE 1960
1930 J1=1
1940 M1=3
1950 RETURN
1960 IF (J1>29)*(M1=2)*(A1=0)THEN 1970 E
LSE 2000
1970 J1=1
1980 M1=3
1990 RETURN
2000 IF (J1>31)*(M1=3)THEN 2010 ELSE 204
0
2010 J1=1
2020 M1=4
2030 RETURN
2040 IF (J1>30)*(M1=4)THEN 2050 ELSE 208
0
2050 J1=1
2060 M1=5
2070 RETURN
2080 IF (J1>31)*(M1=5)THEN 2090 ELSE 212
0
2090 J1=1
2100 M1=6
2110 RETURN
2120 IF (J1>30)*(M1=6)THEN 2130 ELSE 216
0
2130 J1=1
2140 M1=7
2150 RETURN
2160 IF (J1>31)*(M1=7)THEN 2170 ELSE 220
0
2170 M1=8
2180 J1=1
2190 RETURN
2200 IF (J1>31)*(M1=8)THEN 2210 ELSE 224
0
2210 J1=1
2220 M1=9
2230 RETURN
2240 IF (J1>30)*(M1=9)THEN 2250 ELSE 228
0
2250 J1=1
2260 M1=10
2270 RETURN
2280 IF (J1>31)*(M1=10)THEN 2290 ELSE 23
20
2290 J1=1
2300 M1=11
2310 RETURN
2320 IF (J1>30)*(M1=11)THEN 2330 ELSE 23
60
2330 J1=1
2340 M1=12
2350 RETURN
2360 IF (J1>31)*(M1=12)THEN 2370 ELSE 23
90
2370 J1=1
2380 M1=1
2390 RETURN
2400 CALL KEY(0,K,S)
2401 IF S=0 THEN 2400
```

```
2410 INPUT "Voulez_vous un autre bioryth
me(O OÙ N):":R$
2420 IF (R$="O")+(R$="N")THEN 2430 ELSE
2410
2430 IF R$="O" THEN 2440 ELSE 2480
2440 PRINT : :
2450 INPUT "Du meme mode(O OU N):":R$
2460 IF (R$="O")+(R$="N")THEN 2470 ELSE
2450
2470 IF R$="O" THEN 2490 ELSE 370
2480 END
2490 PRINT : :
2500 INPUT "De la meme personne(o ou n):
":R$
2510 IF (R$="O")+(R$="N")THEN 2520 ELSE
2500
2520 IF R$="O" THEN 730 ELSE 640
2530 V=P
2540 CALL CLEAR
2550 FOR I=1 TO BG
2560 IF I>4 THEN 2570 ELSE 2600
2570 CALL SOUND(200,1232,6)
2580 CALL KEY(O,KE,ST)
2590 IF ST=0 THEN 2580
2600 B=23
2610 PRINT "BIORYTHME DU:";J1;"/";M1
2620 FOR J=1 TO 3
2630 GOSUB 1340
2640 B=B+5
2650 PRINT G$(J);":";W
2660 NEXT J
2670 V=V+1
2680 J1=J1+1
2690 IF J1>28 THEN 2700 ELSE 2710
2700 GOSUB 1880
2710 PRINT : :
2720 NEXT I
2730 GOTO 2400
2740 PRINT "PROGRAMME:BIORYTHME": : : :
2750 INPUT "Voulez_vous des explications
(O OU N):?":R$
2760 IF (R$="O")+(R$="N")THEN 2770 ELSE
2740
2770 IF R$="O" THEN 2790 ELSE 2780
2780 RETURN
2790 CALL CLEAR
2800 PRINT "BIORYTHME":"Pour calculer vo
tre bioryt-":"hme:"
2810 PRINT "apres ces explications l'":"
ordinateur va vous demand":"er le mode q
ue vous desirez"
2820 PRINT "(graphique ou numerique).":"
Puis votre date de naissance":"(pour l'a
nnee apres 1500)"
2830 PRINT ",la date de depart pour le "
2840 PRINT "calcul et le nombre de ":"jo
urs a couvrir.":"Le mode graphique est u
ne":"sinusoidale."
2850 PRINT "Si vous demander un bioryth-
":"me de plus de 22 jours il":"suffit d'
appuyer sur une"
2860 PRINT "touche pour avoir le biory-"
:"thme d'un jour de plus et":"ainsi de s
uite."
2870 PRINT "Pour le mode num procedez ":
"de la meme facon."
2880 INPUT "POUR CONTINUER APPUYEZ SUR
<ENTER>:":RR$
2890 RETURN
2900 IF (J<1)+(J>31)+(M<1)+(M>12)+(A<150
0)THEN 2950
2910 IF (J>30)*((M=4)+(M=6)+(M=9)+(M=11)
)THEN 2950
2920 IF (J>29)*(ZA=0)THEN 2950
2930 IF (J>28)*(M=2)*(ZA<>0)THEN 2950
2940 RETURN
2950 INPUT "Date exacte s'il vous plait:
":J,M,A
2960 XA=A/4
2970 YA=INT(XA)
2980 ZA=XA-YA
2990 GOTO 2900
3000 IF N<HH THEN 3020 ELSE 3010
3010 GOTO 860
3020 PRINT "La date exacte s'il vous pla
it!!!"
3030 GOTO 730
```

Economisez de la disquette

Un truc pour les heureux possesseurs d'unités de disquettes. Sur TOUTES les disquettes du commerce, les deux faces sont enduites de matière magnétique. Cependant une seule de ces faces est contrôlée en fabrication et les disquettes sont vendues en "simple face".

Pour les utiliser en double face, il faut en premier lieu faire une encoche sur le côté, symétrique par rapport à celle qui est déjà faite. L'outil idéal est une pince de tiercé. Prenez bien garde à faire cette seconde encoche à la même hauteur que la première. Ne la faites pas trop profonde afin de ne pas abimer le support magnétique.

En second lieu, il faut vous munir d'une perforatrice à un seul trou dont vous garnissez la base d'une matière douce comme de la feutrine adhésive. Cet outil prêt, examinez votre disquette. Lorsque vous la tenez face à vous, trou de lecture vers le bas, vous constatez la présence d'un petit trou sur le côté droit du rond central. Il vous faut faire le même trou mais sur le côté gauche et à même hauteur.

Important : Fabriquer un masque en carton sur lequel vous allez repérer soigneusement les endroits que vous devrez perforer.

Glissez la partie basse de la perforatrice entre le support magnétique et le carton, positionnez bien à l'endroit précis, puis percez. Cette opération est très délicate et le risque est grand de détériorer au moins en partie la disquette.

Nous vous conseillons de ne réaliser cette opération qu'avec des disquettes neuves. En effet, si vous détériorez une partie de disquette sur laquelle se trouve une information, cette information sera perdue à jamais. Si, en revanche, il n'y a rien sur la disquette, l'opération d'initialisation repérera le secteur détérioré comme un secteur occupé. Il n'y aura donc plus jamais d'information à cet endroit. (359*2)

Micro-informations

Les informations publiées ci-dessous nous ont été transmises par Texas Instruments. Nous n'avons pas eu l'occasion, comme ces produits sont pour la plupart, seulement annoncés et non encore disponibles (à la date de rédaction de cet article), de les tester pour vous. Nous ne nous priverons pas de le faire dès qu'ils seront disponibles. Tous les produits présentés ci-dessous sont des produits nouveaux, non encore commercialisés au moment de la rédaction de cet article.

Logiciels co-édités TI - Ivan Berg

Ces logiciels sont attendus sous forme de cassettes, au prix approximatif de 140 francs pièce, et devraient être disponibles dans le courant du mois de novembre.

Tests de maths - 1

Programme de révision en arithmétique, algèbre, géométrie, sur les vecteurs et en trigonométrie.

Tests de maths - 2

Programme permettant de s'exercer dans des domaines variés tels que l'arithmétique, la résolution d'équations, l'étude analytique de fonctions et la trigonométrie.

Tests de physique

Programme éducatif traitant des circuits électriques, des lois de l'électricité et de magnétisme, de l'électronique, de la propagation des ondes, et de la lumière, des forces et des lois de la mécanique.

Tests de chimie

Découvrez les secrets de la classification périodique des éléments, des liaisons atomiques, des niveaux d'énergie, des questions REDOX de la chimie organique.

Tests de biologie - 1

Programme traitant de sujets élémentaires de biologie, nutrition, respiration, circulation, reproduction, système nerveux et biologie de l'environnement.

Tests de biologie - 2

Ce programme constitue la suite logique du volume 1. Il traite des mêmes sujets, mais plus en profondeur.

Logiciels co-édités TI - Collins Education

Ces logiciels, attendus en novembre, seront commercialisés au prix approximatif de 140 francs pièce.

Introduction au TI-99/4A Volume 1

Ce logiciel, destiné aux débutants, est constitué d'un livre décrivant et expliquant les instructions du TI Basic (RUN, SAVE, OLD, GO TO, FOR-NEXT) ainsi que d'une cassette comportant plusieurs programmes (TI Basic) expliquant, entre autres, la programmation de graphiques, de sons et de couleurs ainsi que l'utilisation d'instructions propres au TI Basic telles que CALL HCHAR, CALL KEY...

Introduction au TI-99/4A Volume 2

Ce logiciel est la suite du précédent. Il explique des instructions en TI Basic un peu plus complexes telles que ON.. GOSUB, GOSUB, VAL, STR$, DIM, OPTION BASE. Il traite aussi de la façon d'organiser un programme, des sous-programmes et la définition de caractères. Un programme permet d'illustrer l'utilisation des matrices, du graphisme et des sons.

Les techniques des programmes de jeu - Niveau 1

Ce programme permet d'acquérir l'art et la manière d'écrire des jeux sur ordinateur. Le manuel d'accompagnement présente des techniques de programmation, de déplacement, de labyrinthes, de tirs, de devinettes et de contrôle d'objets par l'intermédiaire du clavier. La cassette contient six programmes en TI Basic illustrant toutes ces techniques.

Les techniques de programmes de jeu - Niveau 2

Ce programme comporte un livre traitant des jeux de stratégie et insistant sur le développement de sous-programmes permettant à l'ordinateur de "jouer intelligemment".

Les programmes stockés sur la cassette (Mastermind, Bataille Navale, Poker, etc.) sont des programmes de jeu en eux-mêmes et servent d'illustration pour les techniques de programmation de jeux.

Introduction aux échecs

La cassette contient dix programmes permettant d'apprendre le mouvement des pièces, leur valeur et leur tactique de déplacement. Ce programme permet de jouer une partie contre l'ordinateur ou tout autre adversaire. Il existe même une option permettant au joueur de battre l'ordinateur ! Ce programme est vraiment conçu pour ceux qui veulent apprendre à jouer aux échecs.

Technique de gestion de données

Le manuel d'accompagnement explique en détail les techniques de récupération de données et montre comment les programmes stockés sur la cassette peuvent être utilisés pour la gestion et l'analyse de données.

Editeur VIFI International

Les deux produits décrits ci-dessous seront réalisés sous la forme de cassettes et vendus au prix approximatif de 100 francs pièce à partir du mois d'octobre.

Compléments et multiples

Cassette éducative s'adressant à un, deux ou trois enfants, avec trois niveaux de difficulté, et comportant deux jeux. Le principe du jeu "Complément" consiste à assortir deux cartes complémentaires dont la somme, selon le niveau choisi, est égale à 50, 100 ou 1000. Dans le jeu "Multiples", il s'agit de reconnaître deux cartes multiples l'une de l'autre qui, selon le niveau de difficulté, doivent être le double, le triple ou le quadruple de l'autre. Ces deux jeux permettent de se familiariser avec des notions importantes.

Mes premiers mots croisés

Cette cassette correspond au jeu bien connu des cruciverbistes. Adaptée à l'enfant, elle lui fait faire des exercices de vocabulaire. Pour les mots difficiles, l'enfant peut demander une deuxième définition. Le contrôle du mot tapé dans la grille s'effectue lettre par lettre, ce qui constitue une bonne aide orthographique. Cette cassette comporte également un programme permettant à l'utilisateur de construire ses propres mots croisés.

Editeur Magnard

Les trois progiciels présentés ci-dessous, baptisés "studiciels" par le fabricant, seront vendus dès le mois d'octobre, sous forme de cassettes et au prix approximatif de 100 francs pièce. Outre ces trois cassettes, de nombreux autres programmes de mathématiques et de grammaire sont prévus pour des échéances ultérieures.

Ponctuation

Quelle impression désagréable laisse une copie ou une lettre maladroitement ponctuée ! Ce studiciel remet à leur juste place tous ces petits signes "! ? : ; , ..." dont le bon usage est

essentiel à la clarté et à la force de tout message.
NDLR ; excellente, idée ?

Ne confondez pas

Savez-vous distinguer et utiliser adéquatement les termes tels que "peux, peut, peu", ou "son, sont", c'est-à-dire des homonymes. Ce studiciel vous apprend à lever de telles ambiguïtés, source de grossières fautes d'orthographe. Il conduit à une meilleure maîtrise de la langue écrite.

Diviseur, PGCD, PPCM

Dans l'apprentissage des mathématiques, PGCD et PPCM sont deux verrous qu'il faut faire sauter. Dans ce studiciel, les nombres deviennent vivants : la dynamique même du travail avec l'ordinateur vous guidera vers le succès.

Cartouches TI

Les quatre cartouches ci-dessous sont attendues pour la fin de l'année et devraient coûter environ 350 francs pièce.

Moonmine

Echappez-vous dans l'espace et donnez-vous des sueurs froides en vous retrouvant aux commandes d'un vaisseau lunaire. Votre mission consiste à récupérer des trésors terrestres qui vous ont été dérobés par le fameux "zigonaute" et ses amis. Le synthétiseur de parole vous permettra de profiter au mieux de ce jeu.

Sneggit

Vous désirez ramasser les œufs qui se trouvent dans votre poulailler. Malheureusement, celui-ci est soudainement envahi par des serpents qui dévorent ces œufs et s'attaquent à vous. Débrouillez-vous pour récupérer le plus grand nombre possible d'œufs.

MASH

Votre mission consiste à ramener au camp MASH 4077 des soldats blessés au cours d'une bataille. Il faut faire vite car ces blessés sont vraiment mal en point. Un aide vigilante vous aidera à optimiser votre score.

Jawbreaker

Cela n'a jamais été aussi amusant d'aller chez le marchand de bonbons, mais attention à l'arracheur de dents...

Cartouches en co-édition

Les cartouches décrites ici sont coéditées par Texas Instruments et un autre éditeur.

Budget familial

Ce programme a été développé par TI en collaboration avec le groupe Expansion (co-éditeur) et Christiane Collange. Il permet de gérer les comptes d'une famille : pour cela, l'utilisateur peut définir jusqu'à 50 catégories de revenus et dépenses. Il peut aussi contrôler chaque élément de revenu ou de dépense, et le comparer à ce qu'il avait initialement prévu et noté, à l'aide d'un tableau et d'un graphisme sophistiqués. Cette analyse peut se faire par mois, par catégorie, ou globalement. Les données introduites par l'utilisateur peuvent être mémorisées sur tout périphérique compatible avec le TI 99/4A (magnétophone à cassette, lecteur de disquette...). L'utilisation de ce programme est extrêmement simple, les textes apparaissant sur l'écran étant rédigés en français. Prévu en fin d'année à 500 francs.

TI-CALC

Co-édité avec Bond Associates Ltd et attendu en fin d'année à 500 francs, TI-CALC est une sorte de feuille de calcul électronique sous forme de cartouche. Ce programme fonctionne avec la console de base et peut utiliser une cassette audio ou une disquette pour stocker des données.

Il peut traiter 6000 éléments sur un nombre variable de colonnes et de lignes. Il comporte les fonctions classiques (insertion ou suppression de lignes ou de colonnes, décomposition de l'écran...). Il permet d'effectuer de nombreux calculs et d'imprimer des rapports au choix de l'utilisateur. Il sera disponible en français, allemand et anglais.

SCRABBLE

Cette cartouche transforme votre TI 99/4A en partenaire pour une partie de Scrabble ! Jusqu'à quatre joueurs (dont trois au maximum sont remplacés par l'ordinateur) peuvent prendre part à une partie. Le TI 99/4A devient alors un redoutable adversaire. Il connait plus de 7000 mots français avec, en outre, différentes variations. La version française du Scrabble est une exclusivité Texas Instruments. Prix : 350 francs, en fin d'année.

TI-LOGO 2

Ce langage, plus spécialement destiné aux enfants, permet de faire comprendre la philosophie informatique tout en laissant à l'utilisateur la maîtrise de l'ordinateur. Il est basé sur le principe philosophique d'éducation (développé par Seymour Papert et l'équipe du laboratoire d'intelligence artificielle du MIT) qui crée un environnement informatique dans lequel les mathématiques et d'autres disciplines d'enseignement formel peuvent être apprises d'une manière naturelle.

Ce sont en fait les utilisateurs qui enseignent à l'ordinateur la manière de procéder. Par rapport au TI LOGO, le TI LOGO 2 dispose de capacités musicales, de lutins plus grands et d'une capacité mémoire utilisateur double (16K).

TI LOGO 2 nécessite l'extension mémoire 32 K. La version française a été développée en collaboration avec des Professeurs des Universités Paris 6 et Paris 7. Cette cartouche devrait être disponible à la date de parution de "99 magazine", pour moins de 1000 francs.

Nouveaux accessoires

Deux nouveaux accessoires sont à présent disponibles. Le magnétophone à cassettes Texas Instruments, avec câble de liaison incorporé et préréglage du volume et de la tonalité est vendu environ 500 francs.

Un modulateur SECAM adaptable sur tout téléviseur français est maintenant commercialisé ; son prix public approximatif est de 600 francs.

Publications en français

De nombreux ouvrages en français sont à présent disponibles pour les utilisateurs du TI 99/4A. Nous ne manquerons pas de commencer à les analyser pour vous dès le numéro prochain. En voici la liste :

- Pratique de l'ordinateur familial TI 99/4A - Editions Radio
- La découverte du TI 99/4A Editions du PSI
- Exercices pour le TI 99/4A Editions du PSI
- Le TI 99/4A à l'affiche Editions du PSI
- La conduite du TI 99/4A Editions Eyrolles
- Jeux et programmes pour l'ordinateur familial Texas Instruments - Shift Editions

NDLR - Après vous avoir parlé de toutes ces cassettes et de toutes ces cartouches qui sont déjà sur le marché ou vont y arriver dans un avenir proche, nous ne pouvons pas passer sous silence le fait que, comme vous avez déjà pu le constater, c'est certainement "99 magazine" qui offre le meilleur rapport performance/prix avec ses cassettes à 55 francs, et même à moins de 50 francs sur abonnement !

Courrier des lecteurs

Nous sommes une association loi de 1901 pour la formation dans la région de Toulon. Notre but étant de réaliser des montages pédagogiques, nous voudrions composer des titres à l'aide du TI-99.

Pour cela, nous souhaiterions savoir :

- Quels programmes enregistrer pour composer des lettres majuscules ou des titres variés;

- Comment connecter une sortie audio et vidéo que l'on peut brancher sur un magnétoscope.

Christian Sagot

La composition de titres n'est sans doute pas tout ce que vous attendez de votre TI-99. Cependant, cette question nous semble importante : il est en effet essentiel, surtout dans des applications de caractère pédagogique, que la présentation soit impeccable.

Le moyen traditionnel pour composer des gros caractères consiste à utiliser l'instruction CALL CHAR(code, "forme").

Le code est le code ASCII du caractère dont vous redéfinissez la forme (reportez-vous au manuel du Basic pour la liste de ces codes).

La forme est le code hexadécimal des caractères binaires composant le dessin.

L'instruction CALL CHAR est l'un des atouts majeurs du TI-99. Nous ne saurions en conséquence trop vous conseiller de rouvrir périodiquement le manuel du Basic pour bien comprendre les principes de son fonctionnement.

Un autre problème pouvant se poser est le suivant : comment éviter ce mouvement vers le haut lors de l'apparition de textes à l'écran ?

Le petit programme ci-dessous résout cette question. Le principe en est très simple : lorsqu'un programme écrit en Basic simple commence à tourner, l'écran devient vert clair. Sauf ordre contraire (par l'instruction CALL SCREEN), il gardera cette couleur jusqu'à ce que l'application soit interrompue.

```
100 CALL CLEAR
110 GOSUB 300
120 PRINT "  ICI PREND PLACE LE TEXTE"
130 PRINT "  QUE VOUS VOULEZ FAIRE"
140 PRINT "  APPARAITRE AU MILIEU"
150 PRINT "  DE L'ECRAN.":::::::::
160 PRINT "UNE TOUCHE POUR FINIR"::
170 GOSUB 370
180 CALL KEY(0,K,S)
190 IF S=0 THEN 180
200 END
210 REM ============================
220 REM  SUITE DU PROGRAMME
230 REM ============================
240 REM ........
250 REM ........
260 REM ........
270 REM ============================
280 REM  ROUTINE DE TRANSPARENCE
290 REM ============================
300 FOR SET=1 TO 16
310 CALL COLOR(SET,1,1)
320 NEXT SET
330 RETURN
340 REM ============================
350 REM  APPARITION DU TEXTE
360 REM ============================
370 FOR SET=1 TO 16
380 CALL COLOR(SET,2,1)
390 NEXT SET
400 RETURN
```

Avant de composer un écran, nous faisons passer le programme par une routine de transparence : tous les caractères deviennent verts sur fond vert; autrement dit, on ne les voit pas lorsqu'ils effectuent leur mouvement de montée (lignes 250 à 280).

Nous donnons ensuite la série des ordres PRINT composant l'écran, puis nous envoyons le programme vers une autre routine qui modifie la couleur des caractères, par exemple en noir sur fond vert (lignes 320 à 350). De cette manière, les caractères n'apparaissent à l'utilisateur qu'une fois qu'ils occupent leur place dans l'écran et on évite d'attraper le mal de mer en les regardant monter.

L'écran que vous présenterez de cette manière peut être n'importe quoi : du texte, du dessin, des informations chiffrées, etc. Mais cette nouvelle procédure oblige à adopter une nouvelle philosophie consistant à travailler par pages d'écran : lorsqu'un écran est utilisé, on l'efface par un CALL CLEAR, on envoie le programme à la procédure de transparence, on compose le nouvel écran et, enfin, on renvoie à la procédure de réapparition.

Pour ce qui est de brancher votre écran avec un magnétoscope, il vaudrait mieux poser la question à un spécialiste de la vidéo. En revanche, si vous trouvez une solution simple pour réaliser cette connection, nombre de nos lecteurs seront intéressés. N'hésitez donc pas à nous l'envoyer.

Bravo à tous pour votre magazine, il est vraiment très bien. Les "texans" (les utilisateurs de Texas ne sont-ils pas des texans ?) ont enfin leur magazine. Ca fait plaisir de voir que l'on pense à nous.

Je voudrais vous signaler quelques erreurs dans vos programmes :

- Les Tours de Hanoï : la ligne 830 passe très mal et lorsque l'on lance le programme, à la deuxième tentative de changement de disque, le programme refuse d'aller plus loin et se bloque sur cette ligne. Pourriez-vous m'indiquer la rectification à faire ?

- Le Dernier Robot : cette fois, c'est au début que le programme se bloque, précisément à la ligne 220. Et là, rien à faire : j'ai tout essayé, mais étant débutant, je n'ai pu voir le "hic". Pouvez-vous m'aider ?

Laurent Pecheux

Les listings que nous imprimons dans "99" sont directement issus d'un TI-99 sur lequel ils ont tourné et tournent encore sans problème. Il serait peut-être intéressant que vous nous indiquiez la nature des messages d'erreur que vous obtenez.

Sachez cependant que les numéros de lignes contenus dans les messages d'erreur ne sont pas forcément ceux où se situe l'erreur causant leur apparition. Une erreur peut très bien se situer ailleurs mais rendre impossible l'exécution d'une instruction ailleurs dans le programme.

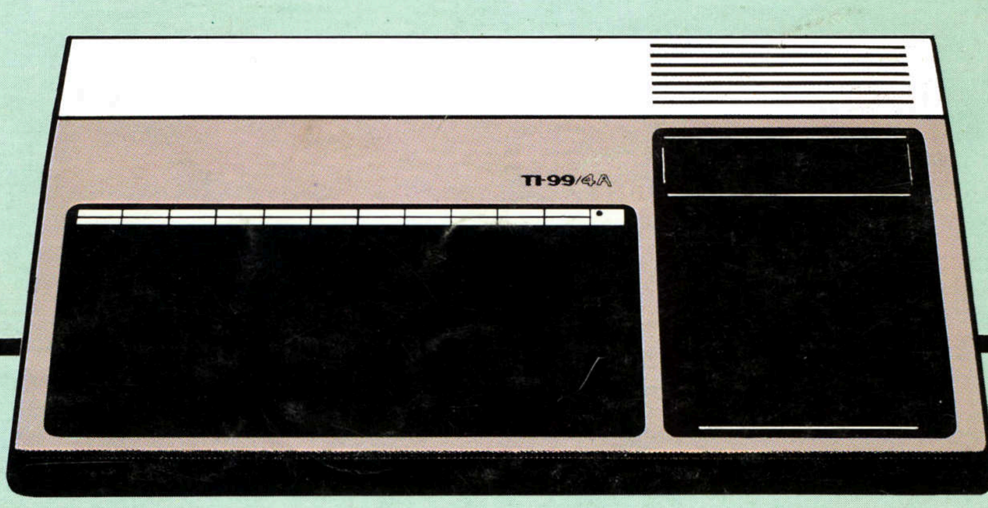
La relecture d'un programme est un travail délicat et d'autant plus fastidieux que ce programme est long. Nous vous conseillons vivement de vous faire aider d'un camarade ou d'un membre de votre famille pour le faire : vous relisez le programme à l'écran, à haute voix, pendant que votre collaborateur vérifie sur le listing (celui qui est dans la revue, et non celui que vous pouvez avoir édité vous même).

Nos programmes étant bons, nous ne voyons que cette solution pour vous aider. La solution la plus sûre consiste aussi à acquérir la cassette d'accompagnement de "99" que nous vendons a un prix défiant toute concurrence : tous ces programmes pour 55 FF !

CASSETTE D'ACCOMPAGNEMENT DISPONIBLE

99 MAGAZINE

LE MASTERMIND EN ASSEMBLEUR
CREEZ VOTRE TOUCHE « RESET »
LE BASIC EN PROGRAMMANT
AWARI, CALCUL, PUZZLE,...



NUMÉRO 3 - Trimestriel - Décembre 1983 - 40 F

ISSN 0759-9005

99 MAGAZINE

Sommaire

	Page
Editorial par Hervé Thiriez	5
Un programme de calcul par Pascal Peitrequin	6
Annonce de Texas Instruments	10
Othello à l'essai par Gérard Santraille	10
Le Basic en programmant par Roger Didi	10
Introduction au Pascal par Gérard Santraille	17
Puissance 4 par Daniel Massé	18
Les modes d'adressage en assembleur par Jean-Luc Bazanegue	21
Géographie : la carte de France par Laurent Desobelle	23
Créez votre touche RESET par Jean-Luc Bazanegue	28
Puzzle par Marianne Sutz	30
Awari par Julien Thomas	35
Le sapin de Noël par Gilbert Arribet	38
PARSEC à l'essai par Marianne Sutz	41
Logo, seulement un langage pour enfants ? par Sophie	42
Le Mastermind en assembleur par Jean-Luc Bazanegue	48
Du nouveau sur les cassettes par Gérard Michel	61
Le magnétophone TI par Julien Thomas	61
Bibliographie par Gérard Santraille	62
A vous de programmer	62
Courrier des lecteurs par Alexandre Duback	64
Erratum et annonces	65
Courrier des clubs	65
Trucs et astuces	9 - 34 - 50 - 51 - 53 - 54

Les annonceurs
L'Ordinateur Individuel : page 2 - Socrelec : page 46 - Goldy : page 63 - PSI Diffusion : page 67 - Texas Instruments : page 68.

Editions MEV - 49, rue Lamartine - 78000 Versailles - Tél. : (3) 951.24.43

(Dépôt légal : 4e trimestre 1983)

Editorial

Nous avons tous été secoués par LA NOUVELLE, apprenant avec surprise, pour ne pas dire stupeur, que Texas Instruments arrêtait de fabriquer le TI-99/4A. C'est venu tellement vite que les responsables européens de Texas n'ont eux-mêmes été avertis que 48 heures avant les journaux ! On savait que, depuis quelque temps, Texas perdait de l'argent dans l'informatique individuelle, mais, comme le dit Contact, la revue de la FNAC (numéro de décembre), "le TI 99 n'a toujours pas de concurrent sur le marché français". C'est toujours vrai si l'on considère le nombre de livres et de programmes écrits sur le sujet, ainsi que l'existence de la revue spécialisée que vous nous faites l'honneur de lire ...

Ceci dit, Texas s'engage à alimenter le parc en périphériques et logiciel pendant encore plusieurs mois, et à assurer la maintenance pendant plusieurs années. Pour sa part, 99 magazine a décidé de ne pas "abandonner" tous ceux qui, comme vous, lui ont accordé leur confiance. En tout état de cause, nous continuons la revue.

Nous prions nos lecteurs de nous excuser pour les débuts un peu difficiles de la revue, dont la qualité n'était pas à la hauteur de ses ambitions. Nous avions placé notre confiance dans un rédacteur qui, comme le prouve l'erratum publié dans ce numéro, n'a pas fait preuve de toute la conscience professionnelle à laquelle on pouvait s'attendre. Nous demandons aux lecteurs qui nous ont écrit jusqu'en septembre de nous pardonner si leurs lettres sont restées sans réponse : de nombreuses informations nous ont fait défaut après le départ de ce collaborateur. Heureusement, il a été remplacé par Jean-Luc Bazanegue, assisté des conseils de Gérard Santraille : vous pourrez constater le progrès de la revue dans ce numéro 3.

*Dans ce numéro, **Gérard Santraille** vous propose une introduction au Pascal et vous donne, dans la bibliographie, son avis sur plusieurs ouvrages. Il analyse également pour vous la cartouche Othello. **Julien Thomas** a testé le magnétophone à cassettes Texas Instruments et commente le programme Awari qui nous vient de nos amis du club TISOFT-HOME. **Roger Didi** vous propose un second article d'initiation au Basic. **Sophie,** pour sa part, continue son voyage au pays de la tortue avec son article sur Logo.*

***Laurent Desobelle** apporte sa contribution avec son programme Géographie, et **Pascal Peitrequin** développe vos capacités en calcul mental grâce au programme Calcul. **Daniel Massé** vous offre un excellent Puissance 4, tandis que **Gilbert Arribet** vous présente un magnifique sapin de Noël. **Jean-Luc Bazanegue** passe en revue les modes d'adressage du micro-processeur TMS-9900, déploie toute sa compétence en la matière avec un Mastermind en assembleur, et indique comment installer une touche "RESET" sur le TI-99/4A. **Marianne Sutz** vous propose un programme de puzzle sur ordinateur ainsi que l'essai de la cartouche Parsec. Enfin, **Alexandre Duback** assume la lourde tâche du Courrier des Lecteurs.*

Nous attendons avec impatience vos programmes originaux : envoyez-les sur cassette ou disquette. Les programmes publiés sont rémunérés à raison de 100 francs par page, sans compter la célébrité immédiate que cela vous apporte ! Excusez-nous, mais le prix de 150 francs par page indiqué dans le numéro précédent était l'une des erreurs provoquées par les conditions aberrantes dans lesquelles nous avons été forcés de travailler à la sortie du numéro 2.

Nous sommes désolés des troubles issus des grèves des PTT. Nombreux sont les abonnés qui ont reçu leur numéro très tard. Certains même ne l'ont pas encore reçu. Il nous est difficile d'assumer les conséquences de cette grève à laquelle nous ne pouvons rien. Toutefois, en signe de bonne volonté, nous sommes prêts à envoyer à tout abonné le numéro (et la cassette) qu'il n'aurait pas reçu, contre un chèque pour 50 % du montant correspondant.

Plus le nombre d'abonnés sera élevé, meilleur sera le service que nous pourrons vous fournir. N'oubliez pas qu'un abonnement d'un an à 99 magazine avec ses cassettes d'accompagnement vous offre, pour le prix d'une cartouche, de nombreux conseils et de multiples programmes. Abonnez-vous !

Hervé Thiriez

Rédacteur en chef - Directeur de la publication : Hervé Thiriez. **Comité de rédaction :** Jean-Luc Bazanegue, Thomas Coppens, Roger Didi, Gérard Santraille. **Dessins :** Laurent Bidot. **Ont collaboré à ce numéro :** Gilbert Arribet, Jean-Luc Bazanegue, Laurent Desobelle, Roger Didi, Alexandre Duback, Daniel Massé, Gérard Michel, Gérard Santraille, Sophie, Marianne Sutz, Julien Thomas.

Editions MEV - 49, rue Lamartine - 78000 Versailles. Tél. : (3) 951.24.43.

Régie Publicitaire : Force 7, Anne Jourdan, 39, rue de la Grande-aux-Belles, 75483 Paris Cedex 10. Tél. : (1) 238.66.10.

Diffusion auprès des boutiques et libraires : Editions du PSI, 44-51, rue Jacquard, BP 86, 77400 Lagny-sur-Marne.

Composition : Télécompo, 13-15, avenue du Petit Parc, 94300 Vincennes.

Impression : Imprimerie Rosay, 47, avenue de Paris, 94300 Vincennes. Tél. : (1) 328.18.63.

Calcul

Pascal Peitrequin

Ce programme en **Basic Etendu** est un jeu éducatif ayant pour base le calcul mental. Ce programme peut certainement aider les enfants qui ne sont pas spécialement attirés par les mathématiques et constitue un exemple concret d'utilisation du TI-99/4A, et de l'informatique en général, à des fins pédagogiques.

De nombreuses options et niveaux de jeu sont disponibles, et cela sans nuire à la facilité d'utilisation du programme. Des commandes sans ambiguïté étaient indispensables puisque ce programme doit pouvoir être utilisé par de jeunes enfants.

Le nombre de joueurs peut dépasser 25 ; il est donc possible de faire jouer (ou travailler ?) une classe entière.

NDLR : Nous remercions notre lecteur pour son excellent travail et encourageons les autres lecteurs ayant écrit des programmes intéressants à nous les faire parvenir.

```
100 REM ***************
110 REM *    CALCUL    *
120 REM ***************
130 REM * BASIC ETENDU *
140 REM ***************
150 REM *  COPYRIGHT   *
160 REM *    PASCAL    *
170 REM *  PEITREQUIN  *
180 REM *      ET      *
190 REM * 99  MAGAZINE *
200 REM ***************
210 REM MODIFICATION EVENTUELLE*********
*****************
220 REM CHANGER LA VARIABLE "FLE" POUR M
ODIFIER LA VITESSE DE LA FLECHE(LIGNES 4
00 A 450)
230 ON WARNING NEXT
240 CALL CLEAR :: DIM A$(20):: DIM SC(20
):: DIM O(48):: CALL SCREEN(16)
250 DISPLAY AT(9,1):"E C O L E S  DE   P
 U L L Y"
260 FOR D=1 TO 50 :: NEXT D
270 DISPLAY AT(11,1)BEEP:"JEU DE CALCUL
No 5  5/1983"
280 GOSUB 2550 !LECTURE DES DATA(NOTES)
290 GOSUB 2510
300 CALL SCREEN(4)
310 INPUT "   NOMBRE DE JOUEURS ? ":J
320 CALL SOUND(200,-1,0)
330 CALL CLEAR
340 PRINT TAB(5);"NOMS DES JOUEURS":TAB(
5);"----------------": : :
350 FOR I=1 TO J
360 PRINT "JOUEUR No";I;
370 CALL SOUND(200,400,0)
380 INPUT A$(I):: PRINT :: NEXT I
390 FOR I=1 TO J :: SC(I)=0 :: NEXT I
400 DISPLAY AT(8,4)ERASE ALL:"JEU DE CAL
CUL No 5"
410 DISPLAY AT(11,1):"COMMANDE PAR PRESS
ION? O/N":" ":"Si vous repondez oui,vous
":"utiliserez les boutons de":"tir des m
anettes ou bien leslettres Q,Y"
420 DISPLAY AT(16,1)BEEP:"les lettres Q,
Y pour choisirla reponse.": :"Sinon, le
choix s'effectue":"par le levier de comm
ande":"ou par les touches E,X,S,D"
430 CALL KEY(5,KJ,SJ)
440 IF SJ=0 THEN 430
450 IF KJ=111 OR KJ=79 THEN HUM=1 :: GOT
O 570 ELSE HUM=0
460 IF KJ<>110 AND KJ<>78 THEN 430
470 CALL SOUND(70,-1,0)
480 DISPLAY AT(11,1)ERASE ALL:"AVEC LES
MANETTES" :: DISPLAY AT(13,1):"DE CONTRO
LE ?     OUI/NON"
490 DISPLAY AT(15,1):"si vous repondez o
ui":"vous utiliserez le levier":"de la c
ommande a distance"
500 DISPLAY AT(18,1)BEEP:"sinon, vous ut
iliserez":"les touches E,X,S,D ":"du cla
vier..."
510 CALL KEY(5,M,N)
520 IF N=0 THEN 510
530 IF M=79 THEN 2460
540 IF M=110 THEN 2490
550 IF M=111 THEN JOY=1 :: GOTO 570
560 IF M=78 THEN JOY=0 ELSE GOTO 510
570 CALL SOUND(70,-1,0)
580 DISPLAY AT(11,1)BEEP ERASE ALL:"RECA
PITULATION DES SCORES "
590 DISPLAY AT(13,1):"A CHAQUE TOUR ?"
600 FOR D=1 TO 100 :: NEXT D :: DISPLAY
AT(15,1):"OUI/NON"
610 CALL KEY(5,RAC,SL)
620 IF SL=0 THEN 610
630 IF RAC=111 OR RAC=79 THEN RECAP=1 ::
 GOTO 660
640 IF RAC=110 OR RAC=78 THEN RECAP=0 EL
SE GOTO 610
650 CALL SOUND(70,-1,0)
660 CALL DELSPRITE(ALL)
670 CALL SCREEN(4)
680 CALL CLEAR :: PRINT "  DEGRE DE DIFF
```

```
ICULTE ?"
690 PRINT :: PRINT :: PRINT "   1. 1-50"
 :: PRINT
700 PRINT "   2. 1-100" :: PRINT
710 PRINT "   3. 1-1000" :: PRINT
720 PRINT "   4. 1-10000" :: PRINT
730 PRINT "   5. 1-100000" :: PRINT
740 PRINT "   6. 2-12 (livrets)": : : ::
 CALL SOUND(100,700,0)
750 CALL KEY(5,K1,S1)
760 IF (S1=0)+(K1<49)+(K1>54)THEN 750
770 CALL SOUND(200,-1,0)
780 PRINT "UN INSTANT SVP..."
790 AFF=1
800 IF K1=49 THEN B1=50 :: B2=20 :: B3=2
0 :: FLE=23 :: GOSUB 1710
810 IF K1=50 THEN B1=100 :: B2=40 :: B3=
40 :: FLE=39 :: GOSUB 1710
820 IF K1=51 THEN B1=1000 :: B2=200 :: B
3=200 :: FLE=140 :: GOSUB 1710
830 IF K1=52 THEN B1=10000 :: B2=2000 ::
 B3=2000 :: FLE=180 :: GOSUB 1710
840 IF K1=53 THEN B1=100000 :: B2=5000 :
: FLE=250 :: B3=5000 :: GOSUB 1710
850 IF K1=54 THEN GOSUB 2390 :: FLE=14
860 GOSUB 2530
870 CALL SCREEN(12)
880 IF HUM=1 THEN 930
890 CALL CHAR(128,"00000000FFFFFFFF")
900 CALL CHAR(136,"000000FFFFFFFFFF")
910 CALL COLOR(14,9,12)
920 CALL COLOR(13,5,12)
930 IF HUM=1 THEN STE=1 ELSE STE=2
940 FOR I=1 TO J STEP STE
950 CALL CLEAR :: RANDOMIZE
960 CH$=STR$(R2)
970 IF LEN(CH$)=2 THEN R=2
980 IF LEN(CH$)=3 THEN R=3
990 IF LEN(CH$)=4 THEN R=4
1000 IF LEN(CH$)=5 THEN R=5
1010 IF LEN(CH$)=6 THEN R=6
1020 IF LEN(CH$)=1 THEN R=1
1030 AL=INT(4*RND)
1040 IF AL=0 THEN AL1=8 :: AL2=13 :: AL3
=14 :: AL4=18 :: Q1=10 :: Q2=1 :: Q3=20
:: Q4=10
1050 IF AL=1 THEN AL1=13 :: AL2=14 :: AL
3=18 :: AL4=8 :: Q1=1 :: Q2=20 :: Q3=10
:: Q4=10
1060 IF AL=2 THEN AL1=14 :: AL2=18 :: AL
3=8 :: AL4=13 :: Q1=20 :: Q2=10 :: Q3=10
 :: Q4=1
1070 IF AL=3 THEN AL1=18 :: AL2=8 :: AL3
=13 :: AL4=14 :: Q1=10 :: Q2=10 :: Q3=1
:: Q4=20
1080 IF HUM=1 AND AL=0 THEN AL1=9 :: AL2
=13 :: AL3=17 :: AL4=21
1090 IF HUM=1 AND AL=1 THEN AL1=13 :: AL
2=17 :: AL3=21 :: AL4=9
1100 IF HUM=1 AND AL=2 THEN AL1=17 :: AL
2=21 :: AL3=9 :: AL4=13
1110 IF HUM=1 AND AL=3 THEN AL1=21 :: AL
2=9 :: AL3=13 :: AL4=17
1120 DISPLAY AT(24,1)BEEP:A$(I)
1130 IF HUM<>1 THEN DISPLAY AT(24,16):A$
(I+1)
1140 IF SOUS=1 THEN DISPLAY AT(1,5)BEEP:
C1;"-";C2;"=" ELSE DISPLAY AT(1,5)BEEP:C
1;"+";C2;"="
1150 IF DIVI=1 THEN DISPLAY AT(1,5)BEEP:
C1;":";C2;"="
1160 IF K1=54 THEN DISPLAY AT(1,5)BEEP:C
1;"x";C2;"="
1170 IF HUM=1 THEN 1310
1180 IF JOY=1 THEN 1270
1190 DISPLAY AT(5,1):"S"
1200 DISPLAY AT(5,3):"D"
1210 DISPLAY AT(4,2):"E"
1220 DISPLAY AT(6,2)BEEP:"X"
1230 DISPLAY AT(5,26):"J"
1240 DISPLAY AT(5,28):"K"
1250 DISPLAY AT(6,27):"M"
1260 DISPLAY AT(4,27):"I"
1270 CALL HCHAR(9,13,128,R)
1280 CALL HCHAR(14,4,128,R)
1290 CALL HCHAR(15,23,128,R)
1300 CALL HCHAR(19,13,128,R)
1310 FOR D=1 TO 90 :: NEXT D
1320 IF HUM=1 THEN Q1,Q2,Q3,Q4=19
1330 DISPLAY AT(AL2,Q2):R2
1340 DISPLAY AT(AL3,Q3):R3
1350 DISPLAY AT(AL4,Q4):R4
1360 DISPLAY AT(AL1,Q1):R1
1370 IF HUM=1 THEN 1650
1380 IF JOY=1 THEN 1520
1390 CALL KEY(0,K,S)
1400 IF S=0 THEN 1390
1410 IF K=69 AND AL=0 THEN 1950
1420 IF K=73 AND AL=0 THEN 2030
1430 IF K=83 AND AL=1 THEN 1950
1440 IF K=74 AND AL=1 THEN 2030
1450 IF K=68 AND AL=2 THEN 1950
1460 IF K=75 AND AL=2 THEN 2030
1470 IF K=88 AND AL=3 THEN 1950
1480 IF K=77 AND AL=3 THEN 2030
1490 IF K=69 OR K=83 OR K=68 OR K=88 THE
N SS1=-1 :: CALL COLOR(13,7,12)
1500 IF K=73 OR K=74 OR K=75 OR K=77 THE
N SS2=-1 :: CALL COLOR(13,7,12)
1510 IF JOY<>1 THEN 1660
1520 CALL JOYST(1,X1,Z1)
1530 CALL JOYST(2,X,Z)
1540 IF X1=0 AND Z1=0 AND X=0 AND Z=0 TH
EN 1520
1550 IF Z=4 AND AL=0 THEN 1950
1560 IF Z1=4 AND AL=0 THEN 2030
1570 IF X=-4 AND AL=1 THEN 1950
1580 IF X1=-4 AND AL=1 THEN 2030
```

```
1590 IF X=4 AND AL=2 THEN 1950
1600 IF X1=4 AND AL=2 THEN 2030
1610 IF Z=-4 AND AL=3 THEN 1950
1620 IF Z1=-4 AND AL=3 THEN 2030
1630 IF Z=4 OR X=-4 OR X=4 OR Z=-4 THEN
SS1=-1 :: CALL COLOR(13,7,12)
1640 IF Z1=4 OR X1=-4 OR X1=4 OR Z1=-4 T
HEN SS2=-1 :: CALL COLOR(13,7,12)
1650 IF HUM=1 THEN GOSUB 2660
1660 GOSUB 2080
1670 IF K1=54 THEN GOSUB 2390 ELSE GOSUB
 1710
1680 NEXT I
1690 IF RECAP=1 THEN 2220
1700 GOTO 930
1710 REM
1720 DIVI=INT(RND*3)
1730 IF DIVI=1 THEN 1850
1740 SOUS=INT(RND*2)
1750 C1=INT(RND*B1)+1 :: C2=INT(RND*B1)+
1
1760 IF SOUS=1 AND C1<C2 THEN 1750
1770 IF SOUS=1 THEN R1=C1-C2 ELSE R1=C1+
C2
1780 R2=R1+INT(RND*B2)+1 :: R3=R1-INT(RN
D*B2)+1
1790 IF R3<1 THEN 1750
1800 R5=(INT(R1/10))*10
1810 RESTE=R1-R5
1820 R4=INT((R1+INT(RND*B3))/10)*10+REST
E
1830 IF R1=R2 OR R1=R3 OR R1=R4 OR R2=R3
 OR R2=R4 OR R3=R4 THEN 1710
1840 GOTO 1940
1850 CX1=(INT(SQR(B1)*RND)+1)+2
1860 C2=(INT(SQR(B1)*RND)+1)+2
1870 C1=CX1*C2
1880 R1=C1/C2
1890 R2=R1+(INT(RND*SQR(B2))+1)
1900 R3=R1-(INT(RND*SQR(B2))+1)
1910 IF R3<0 THEN 1850
1920 R4=R1+(INT(RND*SQR(B3))+1)
1930 IF R1=R2 OR R1=R3 OR R1=R4 OR R2=R3
 OR R2=R4 OR R3=R4 THEN 1850
1940 RETURN
1950 CALL SOUND(100,700,0):: CALL SOUND(
100,200,0):: CALL SOUND(100,400,0):: CAL
L SCREEN(4)
1960 SS1=SS1+2
1970 DISPLAY AT(24,1):A$(I);" +";SS1;"pt
s"
1980 FOR D=1 TO 190 :: NEXT D
1990 CALL SCREEN(12)
2000 IF HUM=1 THEN 2990
2010 GOTO 1660
2020 REM*********************************
***************
2030 CALL SOUND(100,400,0):: CALL SOUND(
100,125,0):: CALL SOUND(100,300,0):: CAL
L SCREEN(8)
2040 SS2=SS2+2
2050 DISPLAY AT(24,1):A$(I+1);" +";SS2;"
pts"
2060 FOR D=1 TO 190 :: NEXT D
2070 CALL SCREEN(12):: GOTO 1660
2080 IF SOUS=1 THEN DISPLAY AT(1,21):C1-
C2 ELSE DISPLAY AT(1,21):C1+C2
2090 IF DIVI=1 THEN DISPLAY AT(1,21):C1/
C2
2100 IF K1=54 THEN DISPLAY AT(1,21)BEEP:
C1*C2
2110 CALL SOUND(360,-4,0)
2120 CALL HCHAR(2,24,136,R)
2130 FOR D=1 TO 50 :: NEXT D
2140 SC(I)=SC(I)+SS1 :: SC(I+1)=SC(I+1)+
SS2
2150 IF HUM<>1 THEN DISPLAY AT(20,8):"sc
ore total" ELSE DISPLAY AT(23,8):"score
total"
2160 IF HUM<>1 THEN DISPLAY AT(22,1):A$(
I):: DISPLAY AT(22,12):SC(I);"PTS" ELSE
DISPLAY AT(24,1):A$(I);" ";SC(I);"pts"
2170 IF HUM=1 THEN 2190
2180 DISPLAY AT(24,1):A$(I+1):: DISPLAY
AT(24,12):SC(I+1);"PTS"
2190 FOR D=1 TO 100 :: NEXT D
2200 SS1=0 :: SS2=0 :: DIVI=0
2210 RETURN
2220 CALL CLEAR :: CALL SCREEN(4):: FOR
V=1 TO 12 :: CALL COLOR(V,2,4):: NEXT V
2230 PRINT TAB(9);"SCORES"
2240 PRINT
2250 FOR I=1 TO J
2260 PRINT TAB(5);A$(I);TAB(16);SC(I)::
NEXT I
2270 PRINT :: PRINT "1=encore un tour" :
: PRINT :: PRINT "2=autre degre de diffi
culte"
2280 PRINT "3=changement de joueurs" ::
PRINT "4=arret du jeu"
2290 CALL KEY(2,TIR,XX):: CALL KEY(0,FIN
,YY)
2300 IF XX=0 AND YY=0 THEN 2290
2310 CALL SOUND(50,500,0)
2320 CALL SCREEN(12):: FOR V=1 TO 12 ::
CALL COLOR(V,2,12):: NEXT V
2330 IF TIR=18 THEN 1700
2340 IF FIN=49 THEN 1700
2350 IF FIN=50 THEN 680
2360 IF FIN=51 THEN 310
2370 IF FIN=52 THEN PRINT "  a bientot..
." :: RESTORE :: GOSUB 2530 :: STOP
2380 GOTO 2290
2390 REM livrets
2400 C1=INT(RND*11)+1 :: C2=INT(RND*11)+
1
2410 R1=C1*C2 :: R2=R1-(INT(RND*5)+1)
2420 IF R2<1 THEN 2400
```

```
2430 R3=R1+(INT(RND*5)+1):: R4=R1+(INT(R
ND*8)+1)
2440 IF R1=R2 OR R1=R3 OR R1=R4 OR R2=R3
 OR R2=R4 OR R3=R4 THEN 2400
2450 RETURN
2460 PRINT "ATTENTION!
la touche alpha lock doit   etre levee"
2470 GOTO 510
2480 REM MUSIQUE
2490 PRINT "ATTENTION!
la touche alpha lock doit   etre enfonce
e"
2500 GOTO 510
2510 CALL MAGNIFY(2)
2520 FOR P=1 TO 10 :: CALL SPRITE(#P,INT
(RND*10)+48,13,136,112,P*INT(RND*4.5)-2.
25+P/2*SGN(RND-.5),P*INT(RND*4.5)-2.25+P
/2*SGN(RND*-.5)):: NEXT P
2530 RESTORE :: REM MUSIQUE
2540 GOTO 2570
2550 FOR W=1 TO 48 :: READ O(W):: NEXT W
2560 RETURN
2570 TEM=110 :: FOR I=1 TO 48 STEP 3
2580 IF I=46 THEN TEM=800
2590 CALL SOUND(TEM,O(I),0,O(I+1),0,O(I+
2),0)
2600 NEXT I
2610 DATA 131,165,196,165,196,523,262,33
0,392,440,698,587
2620 DATA 196,330,262,247,330,262,131,16
5,196,165,196,523
2630 DATA 131,370,880,131,185,494,185,22
0,523,131,370,880
2640 DATA 131,370,1047,131,370,880,392,4
94,1175,196,494,784
2650 RETURN
2660 REM 1 SEUL JOUEUR*****************
***************
2670 CALL CHAR(123,"080CFEFFFE0C0800")
2680 CALL CHAR(129,"080CFEFFFE0C0800")
2690 CALL CHAR(128,"FFFFFFFFFFFFFFFF")
2700 CALL CHAR(136,"0000000000000000")
2710 CALL COLOR(13,3,12)
2720 CALL VCHAR(8,18,128,3)
2730 CALL VCHAR(12,18,128,3)
2740 CALL VCHAR(16,18,128,3)
2750 CALL VCHAR(20,18,128,3)
2760 CALL VCHAR(AL1-1,17,136,3)
2770 REM DEBUT BOUCLE******************
***************
2780 FOR LI=3 TO 23
2790 IF LI>8 THEN 2820
2800 FOR D=1 TO FLE :: NEXT D
2810 CALL SOUND(40,800,5)
2820 CALL SOUND(30,300,0):: CALL HCHAR(L
I,16,123)
2830 CALL HCHAR(LI-1,16,32)
2840 CALL GCHAR(LI,17,MY)
2850 CALL KEY(2,RT,SW)
2860 IF RT=18 THEN CALL SOUND(100,400,0)
:: GOTO 2890
2870 FOR D=1 TO 8 :: NEXT D :: NEXT LI
2880 RETURN
2890 REM SOUS-PROGR. D'ENTREE POUR UN JO
UEUR************************************
****
2900 IF AL=0 AND MY=136 THEN 1950
2910 IF AL=1 AND MY=136 THEN 1950
2920 IF AL=2 AND MY=136 THEN 1950
2930 IF AL=3 AND MY=136 THEN 1950
2940 IF RT=18 THEN SS1=SS1-1
2950 CALL HCHAR(AL1,15,129)
2960 CALL COLOR(14,9,9)
2970 FOR D=1 TO 50 :: NEXT D
2980 CALL COLOR(14,1,1)
2990 RETURN
```

Evaluation de la mémoire disponible en Basic TI

Lors de l'élaboration d'un programme, il est souvent avantageux (pour ne pas dire indispensable) d'évaluer l'espace mémoire disponible, compte tenu de la place prise par le codage du programme et la mémoire allouée aux variables.

Le Basic TI étant dépourvu de l'instruction SIZE, il convient d'insérer les lignes suivantes dans le programme (les numéros de ligne sont choisis arbitrairement et peuvent être modifiés) :

```
1000 I=I+1
1010 GOSUB 1000
```

Ces instructions doivent absolument être exécutées après l'initialisation de toutes les variables. Si tel n'était pas le cas, le résultat serait inexact.

En tapant ensuite RUN, on obtient rapidement MEMORY FULL IN 1010. Sachant que chaque exécution de l'instruction GOSUB requiert 8 octets, la place encore disponible en mémoire vive peut être obtenue en tapant :

```
PRINT 8*I+23
```

Le "+23" correspond à la place prise par les lignes 1000 et 1010. Remarquons cependant que le résultat obtenu est une approximation à plus ou moins 7 octets puisque la mémoire se sature par blocs de 8 octets. Si la variable I n'est pas utilisée dans votre programme, il faut rajouter 8 au nombre d'octets disponibles (réservation mémoire de I lors de l'exécution).

Il est évident que ces deux lignes doivent être ensuite supprimées pour que le programme puisse fonctionner normalement.

Communiqué de Texas Instruments France

Après l'annonce de l'arrêt de la production aux Etats-Unis du TI-99/4A, Texas Instruments tient à rassurer les utilisateurs de cet appareil :

- la réparation hors garantie sera assurée pendant de nombreuses années. Texas Instruments possède un important stock en pièces détachées, pour faire face à toute intervention pendant ce temps ;
- il y a des stocks d'accessoires et de logiciels permettant de faire face à la demande pendant plusieurs mois ;
- la sortie de certains nouveaux logiciels reste programmée : budget privé, studiciels Magnard, TI-Calc ;
- des logiciels sous forme de cassette seront vendus à partir de 60 francs environ, et sous forme de module à partir de 130 francs.

L'Ordinateur Familial TI-99/4A est le seul ordinateur familial offrant tous ces avantages.

Othello à l'essai

Gérard Santraille

Othello est un module créé par Gabriel Industries et commercialisé par Texas Instruments.

De prime abord, ce programme semble d'excellente facture. Le graphisme est soigné, les options de jeu nombreuses et la stratégie convaincante. Voyons ses principales caractéristiques.

Huit niveaux de jeu sont accessibles, le temps de réponse est instantané au niveau 1 et d'environ une minute au niveau 8 (ce qui est parfaitement raisonnable).

Avant d'essayer ce module, je me croyais plutôt bon à l'Othello. Depuis, je suis devenu nettement plus humble. En effet, je n'ai jamais réussi à battre le programme au niveau 6 et j'y parviens assez rarement au niveau 5.

Autre particularité intéressante : il est possible de soumettre à l'ordinateur un problème donné ou de poursuivre une partie interrompue en entrant le "damier" colonne par colonne (ceci s'avère très pratique lorsque la partie s'étale sur plusieurs jours, ce qui n'est pas rare).

Diverses fonctions sont accessibles en cours de jeu par pression de la touche "AID": , changement de niveau, affichage des coups possibles, conseil de l'ordinateur et même permutation des joueurs.

Notons enfin que l'ordinateur peut être un simple arbitre parfaitement impartial puisque , dans cette option, il consentira à vous montrer les coups jouables mais refusera de vous conseiller.

Un très bon logiciel que cet Othello. Je le conseille vivement à tous les amateurs de jeux de réflexion. Ils trouveront en leur TI-99 un adversaire de choix.

Apprendre le Basic en programmant

Roger Didi

Les programmes sont listés pages 14 à 16

Dans le numéro 1 de 99 Magazine, nous avons commencé une initiation au Basic en nous appuyant sur la construction évolutive d'un programme. Cette démarche ayant eu l'agrément de nombreux lecteurs, nous allons poursuivre dans la même voie. Nous commencerons par écrire un caractère sur l'écran, en un endroit choisi, puis nous modifirons ce caractère à volonté.

Nous essaierons ensuite d'écrire un texte, de créer un curseur qui marque l'endroit où le caractère va s'inscrire... Pourquoi alors ne pas demander à la machine de mémoriser le texte écrit, de le restituer sur demande, de permettre des corrections...

Le lecteur persévérant pourra ainsi se "fabriquer" de proche en proche un programme de traitement de textes. En faisant évoluer ce programme, nous étudierons plusieurs modes de travail utilisant des tests logiques et quelques nouvelles instructions offertes par le TI Basic. Nous supposons en général que le lecteur a assimilé le contenu de l'article "initiation au Basic" du numéro 1 de 99 Magazine mais essayons, dans la mesure du possible, de rendre ce nouveau chapitre abordable par des débutants.

Programme 1 : un caractère

L'instruction CALL KEY est certainement plus utilisée que INPUT lorsque l'on souhaite que l'utilisateur du programme ait l'impression de commander par pression sur des touches le fonctionnement de la machine. Rappelons que CALL KEY(0,K,S) provoque la surveillance du clavier et l'inscription de toute action dans les variables K et S.

La variable S est affectée de la valeur 0 (zéro) lorsqu'aucune touche n'est pressée , de la valeur −1 si la même touche reste pressée, et de la valeur 1 si une nouvelle touche est pressée.

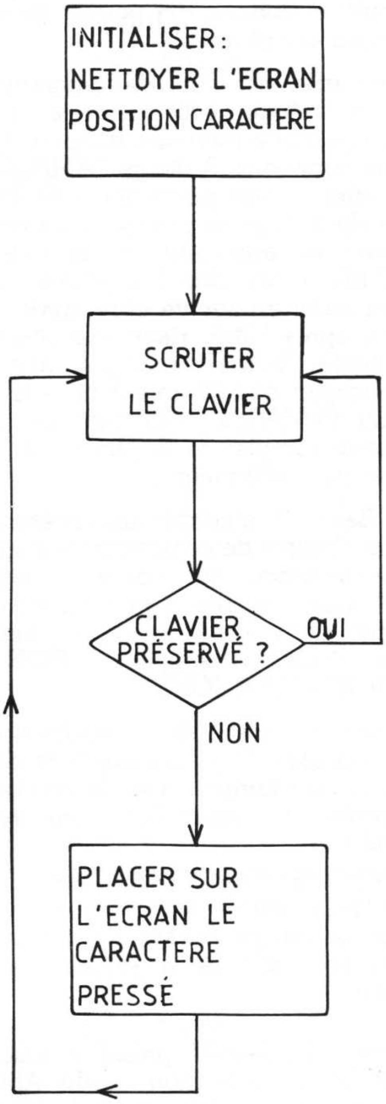
Des tests sur le contenu de cette variable S (dite variable d'ETAT) vont donc permettre de suivre les actions de l'utilisateur du programme sur le clavier.

Lorsqu'une touche est enfoncée, la variable K est affectée du code de la touche pressée et si le premier paramètre du CALL KEY est 0 (zéro), ce code est justement le code ASCII de la touche. Lorsque le clavier n'est pas sollicité, la variable K conserve la valeur −1 (moins 1).

Le programme 1 permet de bien comprendre le fonctionnement de cette instruction. Dès que l'on presse une touche, le caractère correspondant apparaît sur l'écran (ligne 8 et colonne 15). L'organigramme étant fourni, la démarche suivie doit être claire.

Rappelons, en ce qui concerne les organigrammes, que nous plaçons les données, ou les actions, dans des rectangles alors que les tests sont inscrits dans des losanges. Nous choisissons de sortir horizontalement du losange si le test est positif (réponse OUI) et verticalement dans le cas contraire. Les traits fléchés indiquent le sens de lecture de l'organigramme.

Organigramme du Progamme 1

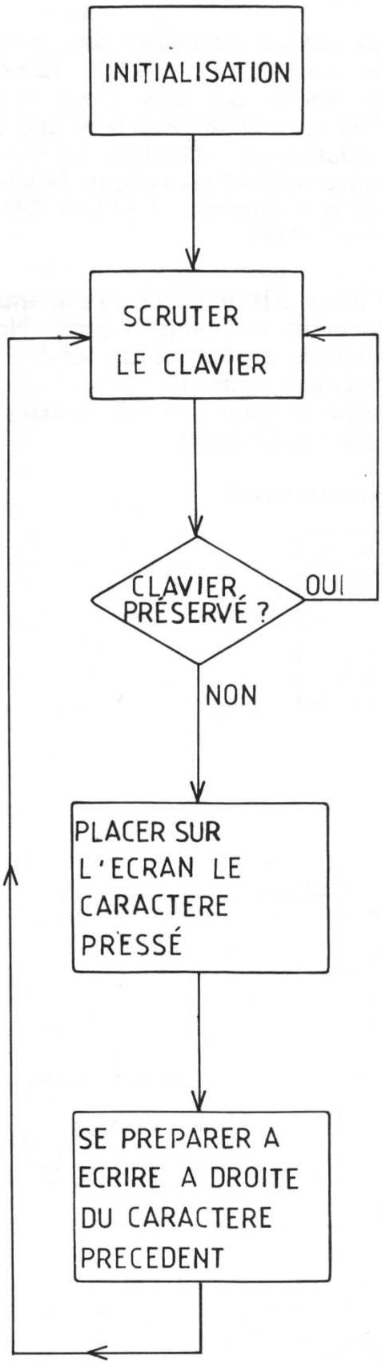


Programme 2 : une suite de caractères

L'instruction CALL HCHAR(Y,X,C) place le caractère dont le code ASCII est C à l'intersection de la ligne N° X avec la colonne N° Y.

Pour que chaque nouvelle lettre se place à la droite de la précédente, et sur la même ligne, il faut donc ne pas changer Y et augmenter X de 1 (on dit incrémenter X de 1) avant chaque appel à CALL HCHAR(Y,X,C). Ainsi, les caractères se plaçant les uns après les autres, il devient possible d'écrire un texte sur l'écran mais... Essayez...

Organigramme du Programme 2



Programme 3 : un curseur clignotant

Le programme précédent présente un premier inconvénient : l'utilisateur ne connaîtra pas l'endroit exact où va s'inscrire le caractère qu'il souhaite écrire ! Nous allons donc l'aider en y plaçant un curseur clignotant. Il faut cependant que ce curseur puisse passer sur des lettres sans les effacer. La démarche est donc la suivante :

1) enregistrer le caractère qui va se trouver sous le curseur

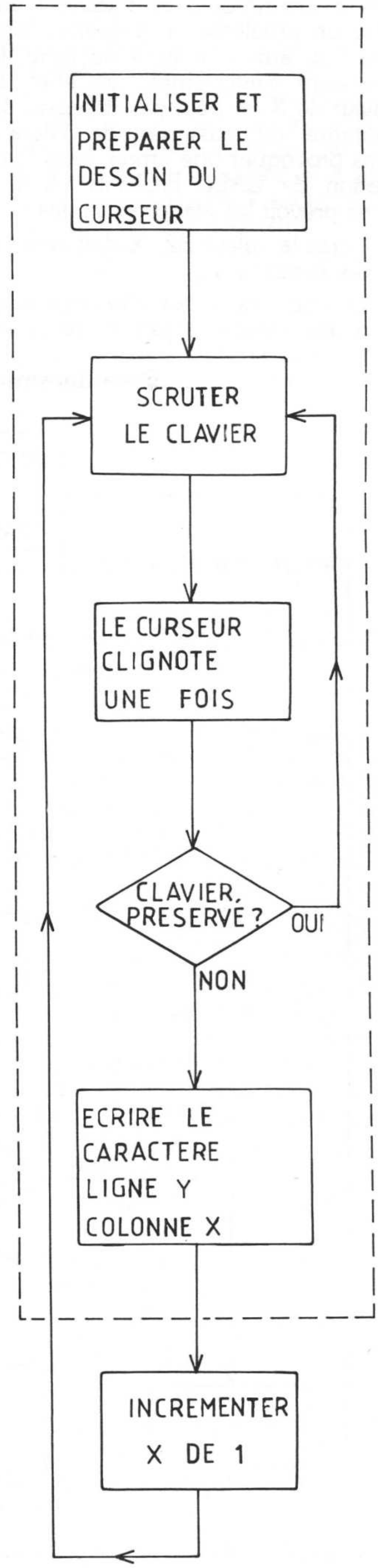
2) dessiner le curseur à cet endroit

3) remettre l'ancien caractère au même endroit

4) recommencer en revenant à l'étape N° 1.

L'instruction CALL GCHAR accomplit justement l'étape No 1. En effet,

Organigramme du Programme 3



CALL GCHAR(Y,X,CAR) affecte la variable CAR du code du caractère situé sur l'écran ligne Y, colonne X : elle permet donc de "lire" sur l'écran. La suite du programme est évidente, l'organigramme montrant où doivent se situer les instructions concernant le curseur.

Programme 4 : occuper tout l'écran

Les programmes 2 et 3 laissent encore un problème en suspens : lorsque l'on arrive en bout de ligne, le processus s'interrompt : en effet, la valeur de X ne peut pas dépasser 32 (nombre de colonnes de l'écran) sans provoquer une erreur dans l'utilisation de CALL HCHAR ! Il faut donc prévoir les étapes suivantes :

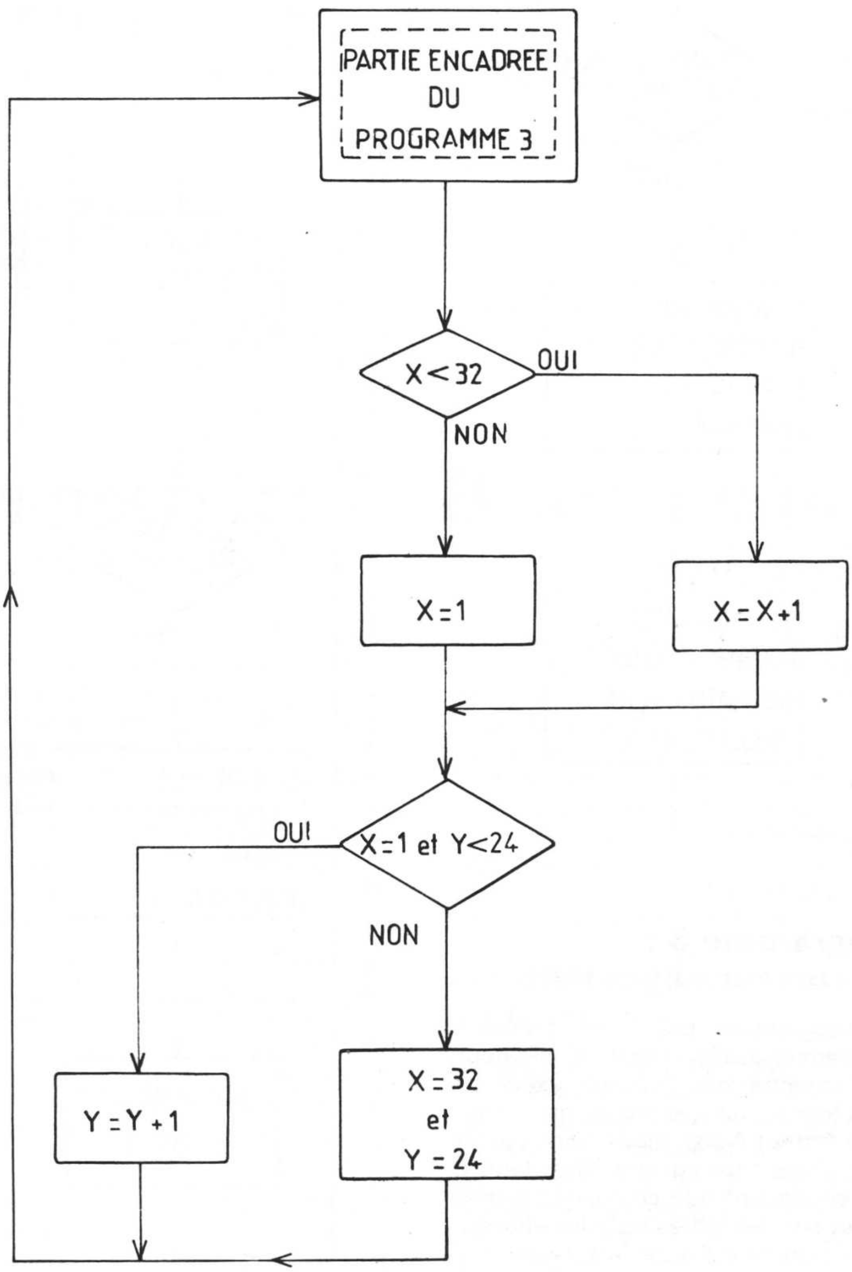
- Après la valeur 32, X doit reprendre la valeur 1 ;
- La valeur de Y doit alors être augmentée d'une unité : on passe en effet à la ligne suivante, mais la valeur de Y ne doit pas dépasser 24.

 Nous avons donc choisi d'arrêter le curseur quand X=32 et Y=24, mais le lecteur pourra modifier ce choix et décider par exemple qu'après X=32 et Y=24, il y aura un CALL CLEAR suivi de X=1 et Y=1.

Nous aurions pu utiliser des instructions du type IF...THEN...ELSE... pour réaliser ces tests. Nous avons préféré en profiter pour faire une démonstration de l'utilisation des calculs logiques utilisant les valeurs de vérité d'une proposition (−1 pour VRAI et 0 pour FAUX).

Les lignes 210 et 220 du programme 4 méritent un temps d'arrêt. Nous conseillons au lecteur de les lire attentivement pour bien comprendre le principe général des calculs utilisant ces valeurs de vérité.

Organigramme du Programme 4



Programme 5 : un beep à chaque lettre

Il peut être utile d'obtenir un beep lorsque la machine est prête à accepter une nouvelle lettre. Construisez un organigramme pour prévoir à quel endroit du programme doit se placer le CALL SOUND correspondant à ce beep.

Programme 6 : mémoriser un texte

Nous voulons ici que la machine mémorise le texte que nous entrons. Nous prévoyons donc de dimensionner un tableau de 100 lignes nommé PHRASE$: ce programme pourra enregistrer 100 phrases distinctes. Un compteur nommé PH permet de numéroter ces phrases.

Nous utilisons dans ce programme une technique particulière : la construction d'une mémoire tampon, que nous nommons d'ailleurs TAMPON$. En effet, il faut permettre à l'utilisateur de corriger sa phrase avant de la valider en appuyant sur la touche ENTER. Cette dernière phrase doit aussi rester en attente et se fabriquer, lettre après lettre, dans une réserve d'attente ("buffer" en anglais) afin de se recopier définitivement dans le tableau PHRASE$ seulement lorsque le texte complet de la phrase est accepté par l'utilisateur.

Le Basic TI autorise une opération entre chaînes de caractères, nommée concaténation, qui utilise le signe "&". Cette opération permet, en accolant deux chaînes, d'en obtenir une seule. Exemple : "BON"& "JOUR"="BONJOUR"

Le tampon désiré se construira donc en concaténant successivement chaque ancien tampon avec le nouveau caractère sur lequel l'utilisateur aura porté son choix.

TAMPON$=TAMPON$&CHR$(C)

CHR$(C) doit évidemment se placer à droite car la phrase se construit, lettre par lettre, de la gauche vers la droite.

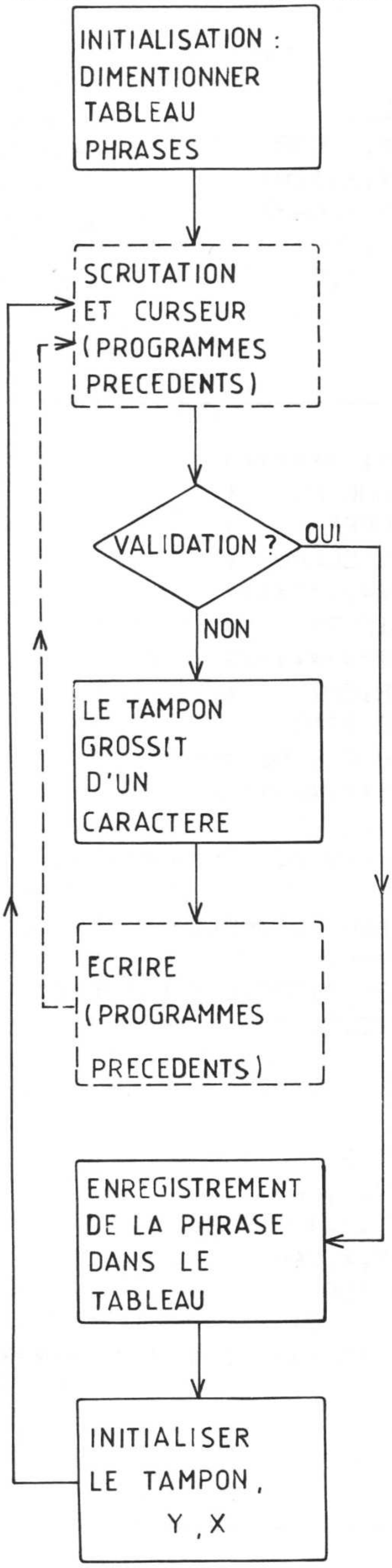
Lorsque l'utilisateur presse la touche ENTER de validation (code ASCII 13), la machine doit accomplir les actions suivantes :

- incrémenter de 1 le compteur de phrases PH ;
- reproduire TAMPON$ dans le tableau PHRASE$;
- renvoyer le curseur à la ligne ;
- revenir à la scrutation de clavier.

Tout ceci est accompli dans les lignes 290-340 de ce programme, qui ne doit plus vous présenter de difficulté.

La ligne 330 incrémente le compteur Y des lignes de l'écran en l'arrêtant à 24. Ce programme n'autorise donc que les textes occupant au maximum un écran : le lecteur pourrait étudier des méthodes lui permettant de construire des textes plus longs.

Organigramme du Programme 6



Programme 7 : corriger et reproduire un texte

Dans le cahier des charges, nous lisons maintenant un nouveau souhait du futur utilisateur : pouvoir corriger la dernière phrase introduite avant de la valider et pouvoir reproduire tout le texte à volonté, par simple pression sur la touche "REDO" : FCTN 8.

La touche "flèche vers la gauche" (FCTN S : code 8) est absolument indiquée pour faire reculer le curseur en effaçant sur l'écran et dans le tampon le dernier caractère introduit par l'utilisateur.

Les étapes de correction doivent donc être les suivantes :

- déterminer le nombre de caractères du dernier tampon (instruction LEN) ;
- calculer la nouvelle position du curseur : c'est un calcul de même nature que celui qui lui permet d'avancer et le lecteur devrait s'y attarder un moment ;
- effacer, en y imprimant un blanc (code 32), le dernier caractère de l'écran ;
- extraire du tampon toutes les lettres sauf la dernière pour fabriquer un nouveau tampon : l'instruction SEG$ permet de réaliser cet objectif (se reporter au manuel pour en étudier le fonctionnement). Pour retrancher 1 à la variable T qui contient le nombre de caractères de TAMPON$, nous avons écrit : T+(T>0). Essayez de comprendre les raisons qui nous ont poussé à écrire cela.

Pour reproduire le texte, nous utilisons un nouveau tampon nommé R$. Chaque phrase est reproduite dans ce tampon, puis lue lettre par lettre et reproduite sur l'écran : le programme qui accomplit ces performances (lignes 290 à 410) ne devrait pas présenter de difficulté.

Programme 8 : salutations et signature

L'utilisateur, qui commence à devenir fort exigeant, voudrait éviter d'avoir à écrire certaines phrases qu'il place dans tous ses textes : son adresse, une formule de politesse, le mot signature etc...

Nous lui assurons qu'une simple pression sur la touche FCTN 9 (BACK) lui permettra d'accomplir cette performance.

Nous enregistrons donc en DATA la suite des phrases à réutiliser, dans l'ordre où elles doivent être appelées : en effet, la directive DATA empile les unes sur les autres toutes les données qui la suivent, et l'instruction READ permettra de relire la pile ainsi obtenue en commençant par la première donnée introduite.

Si l'utilisateur presse FCTN 9 (code 15), le programme reproduit dans TAMPON$ une phrase lue dans la pile et enregistre directement ce tampon dans le tableau PHRASE$: en effet, la phrase étant correcte, il est inutile d'attendre une validation. Il faut cependant prévenir l'utilisateur d'une chose : pour voir sur l'écran le résultat de son action, il lui faudra presser la touche FCTN 8 (REDO). Ce programme ne répond certainement pas complètement au cahier des charges d'un client exigeant et mérite de nombreuses améliorations et transformations : voilà donc un sujet de travail pour les apprentis informaticiens qui les amènera à bien se poser les problèmes posés par l'informatisation d'une activité et à bien évaluer les apports du Basic quant à la solution de ces problèmes : nous encourageons les lecteurs à nous encourager si la démarche utilisée dans cet apprentissage du Basic leur convient et même à nous proposer des programmes qu'il voudrait pouvoir entrer en utilisant leur machine.

Trucs et astuces

Magnétophone et fautes de frappe

Lors de l'utilisation d'un magnétocassette pour la sauvegarde et la restitution de données et programmes, il peut arriver que l'on tape un OLD à la place d'un SAVE (ou l'inverse). Pendant que l'ordinateur est encore sur une directive "REWIND CASSETTE TAPE", "PRESS CASSETTE PLAY" ou "PRESS CASSETTE STOP ... THEN PRESS ENTER", il est possible d'interrompre la procédure en cours. En effet, chaque fois que l'on a la main (l'ordinateur attend une réponse), l'action peut être modifiée en appuyant sur E (pour EXIT), C (pour CHECK) ou R (pour READ ou RECORD selon les cas).

- E arrête le processus avec l'affichage de "PRESS CASSETTE STOP" suivi d'un message I/O ERROR sans aucune conséquence.
- C provoque l'affichage de "REWIND CASSETTE TAPE" et lance la routine de vérification du programme.
- R relance la procédure de lecture ou d'écriture selon que l'on a frappé un OLD ou un SAVE.

```
10 REM *****************
20 REM * PROGRAMME No 1 *
30 REM *  UN CARACTERE  *
35 REM *****************
40 REM *    BASIC TI    *
45 REM *****************
50 REM *   COPYRIGHT    *
55 REM *   ROGER DIDI   *
60 REM * ET 99 MAGAZINE *
65 REM *****************
70 REM
95 REM   On scrute l'ecran, on ecrit le
caractere correspondant  a la touche en
(X=15,Y=8).
100 CALL CLEAR
110 X=15
120 Y=8
130 CALL KEY(0,C,S)
140 IF S=0 THEN 130
150 CALL HCHAR(Y,X,C)
160 GOTO 130
```

```
10 REM *****************
15 REM * PROGRAMME No 2 *
20 REM *    SUITE DE    *
25 REM *   CARACTERES   *
30 REM *****************
35 REM *    BASIC TI    *
40 REM *****************
45 REM *   COPYRIGHT    *
50 REM *   ROGER DIDI   *
55 REM * ET 99 MAGAZINE *
60 REM *****************
65 REM
70 REM la variable X augmente de 1 a cha
que tour.
75 REM
100 CALL CLEAR
110 X=15
120 Y=8
130 CALL KEY(0,C,S)
140 IF S=0 THEN 130
150 CALL HCHAR(Y,X,C)
160 X=X+1
170 GOTO 130
```

```
10 REM *****************
15 REM * PROGRAMME No 3 *
20 REM *   UN CURSEUR   *
25 REM *   CLIGNOTANT   *
30 REM *****************
35 REM *    BASIC TI    *
37 REM *****************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM *****************
60 REM
70 REM   Le curseur est dessine  sous le
 numero 120.
75 REM   CALL GCHAR GARDE  dans CAR le c
aractere a remplacer
100 CALL CHAR(120,"183C7EFFFF7E3C18")
110 CALL COLOR(12,9,1)
120 CALL CLEAR
130 X=15
140 Y=8
150 CALL KEY(0,C,S)
160 CALL GCHAR(Y,X,CAR)
170 CALL HCHAR(Y,X,120)
180 CALL HCHAR(Y,X,CAR)
190 IF S=0 THEN 150
200 CALL HCHAR(Y,X,C)
210 X=X+1
220 GOTO 150
```

```
10 REM *****************
15 REM * PROGRAMME No 4 *
20 REM *    OCCUPER     *
25 REM *  TOUT L'ECRAN  *
30 REM *****************
35 REM *    BASIC TI    *
40 REM *****************
45 REM *   COPYRIGHT    *
50 REM *   ROGER DIDI   *
55 REM * ET 99 MAGAZINE *
60 REM *****************
65 REM
70 REM   Y augmente de 1 chauque fois qu
e X vaut 32
75 REM   Le curseur s'arrete en bas et a
 droite de l'ecran.
100 CALL CHAR(120,"183C7EFFFF7E3C18")
110 CALL COLOR(12,9,1)
120 CALL CLEAR
130 X=15
140 Y=8
150 CALL KEY(0,C,S)
160 CALL GCHAR(Y,X,CAR)
170 CALL HCHAR(Y,X,120)
180 CALL HCHAR(Y,X,CAR)
190 IF S=0 THEN 150
200 CALL HCHAR(Y,X,C)
210 X=-(X+1)*(X<32)+(X=32)*(Y<24)+X*(X=3
2)*(Y=24)
220 Y=Y+(X=1)*(Y<24)
230 GOTO 150
```

```
10 REM *****************
15 REM * PROGRAMME No 5 *
20 REM *   UN BEEP A    *
25 REM * CHAQUE LETTRE  *
30 REM *****************
```

```
35 REM *    BASIC TI     *
40 REM *****************
45 REM *   COPYRIGHT     *
50 REM *   ROGER DIDI    *
55 REM * ET 99 MAGAZINE *
60 REM *****************
65 REM
100 CALL CHAR(120,"813C7EFFFF7E3C81")
110 CALL COLOR(12,9,2)
120 CALL CLEAR
130 X=15
140 Y=8
150 CALL SOUND(50,400,2)
160 CALL KEY(0,C,S)
170 CALL GCHAR(Y,X,CAR)
180 CALL HCHAR(Y,X,120)
190 CALL HCHAR(Y,X,CAR)
200 IF S=0 THEN 160
210 CALL HCHAR(Y,X,C)
220 X=-(X+1)*(X<32)+(X=32)*(Y<24)+32*(X=
32)*(Y=24)
230 Y=Y+(X=1)*(Y<24)
240 GOTO 150
```

```
10 REM *****************
15 REM * PROGRAMME No 6 *
20 REM *   MEMORISER     *
25 REM *    UN TEXTE     *
30 REM *****************
35 REM *    BASIC TI     *
40 REM *****************
45 REM *   COPYRIGHT     *
50 REM *   ROGER DIDI    *
55 REM * ET 99 MAGAZINE *
60 REM *****************
65 REM
80 REM Le texte est enregistre  dans une
 suite de phrases.
90 REM  Chaque phrase est formee dans un
 tampon.
95 REM
96 REM
100 DIM PHRASE$(100)
110 CALL CHAR(120,"813C7EFFFF7E3C81")
120 CALL COLOR(12,9,2)
130 CALL CLEAR
140 X=3
150 Y=4
160 PH=0
170 CALL SOUND(50,400,2)
180 CALL KEY(0,C,S)
190 CALL GCHAR(Y,X,CAR)
200 CALL HCHAR(Y,X,120)
210 CALL HCHAR(Y,X,CAR)
220 IF S=0 THEN 180
230 IF C=13 THEN 290
240 TAMPON$=TAMPON$&CHR$(C)
250 CALL HCHAR(Y,X,C)
260 X=-(X+1)*(X<32)-(X=32)
270 Y=Y-(X=1)
280 GOTO 170
290 PH=PH+1
300 PHRASE$(PH)=TAMPON$
310 TAMPON$=""
320 X=1
330 Y=Y-(Y<24)
340 GOTO 170
```

```
10 REM *****************
15 REM * PROGRAMME No 7 *
20 REM *  CORRIGER ET    *
25 REM *   REPRODUIRE    *
30 REM *    UN TEXTE     *
35 REM *****************
40 REM *    BASIC TI     *
45 REM *****************
50 REM *   COPYRIGHT     *
55 REM *   ROGER DIDI    *
60 REM * ET 99 MAGAZINE *
65 REM *****************
70 REM
90 REM FCTN 8 (REDO) provoque la reimpre
ssion de tout le texte sur l'ecran
95 REM
96 REM
100 DIM PHRASE$(100)
110 CALL CHAR(120,"813C7EFFFF7E3C81")
120 CALL COLOR(12,9,2)
130 CALL CLEAR
140 X=3
150 Y=4
160 PH=0
170 CALL SOUND(50,400,2)
180 CALL KEY(0,C,S)
190 CALL GCHAR(Y,X,CAR)
200 CALL HCHAR(Y,X,120)
210 CALL HCHAR(Y,X,CAR)
220 IF S=0 THEN 180
230 IF C=6 THEN 290
240 IF C=8 THEN 460
250 IF C=13 THEN 530
260 TAMPON$=TAMPON$&CHR$(C)
270 GOSUB 420
280 GOTO 170
285 REM ==================
286 REM REPRODUIRE LE TEXTE
287 REM ==================
290 CALL CLEAR
300 X=3
310 Y=4
320 FOR I=1 TO PH
330 R$=PHRASE$(I)
340 FOR J=1 TO LEN(R$)
350 C=ASC(SEG$(R$,J,1))
360 GOSUB 420
370 NEXT J
380 Y=Y-(Y<24)
```

```
390 X=1
400 NEXT I
410 GOTO 170
415 REM ===================
416 REM LE CURSEUR AVANCE
417 REM ===================
420 CALL HCHAR(Y,X,C)
430 X=-(X+1)*(X<32)-(X=32)
440 Y=Y-(X=1)
450 RETURN
455 REM ===================
456 REM LE CURSEUR RECULE
457 REM ===================
460 T=LEN(TAMPON$)
470 X=((X-1)*(X>1)+32*(X=1))*(T<>1)-X*(T
=1)
480 Y=-(T<>1)*(Y+(X=32))-Y*(T=1)
490 CALL HCHAR(Y,X,32)
500 REM
510 TAMPON$=SEG$(TAMPON$,1,T+(T>0))
520 GOTO 170
525 REM ===================
526 REM PHRASE ENREGISTREE
527 REM ===================
530 PH=PH+1
540 PHRASE$(PH)=TAMPON$
550 TAMPON$=""
560 X=1
570 Y=Y-(Y<24)
580 GOTO 170
```

```
10 REM *****************
15 REM * PROGRAMME No 8 *
20 REM * SALUTATIONS ET *
25 REM *   SIGNATURE    *
30 REM *****************
35 REM *    BASIC TI    *
40 REM *****************
45 REM *   COPYRIGHT    *
50 REM *   ROGER DIDI   *
55 REM * ET 99 MAGAZINE *
60 REM *****************
65 REM
90 REM  FCTN 9 (BACK) provoque l'ecritur
e d'une formule de politesse puis de 'si
gnature'.
95 REM
96 REM
100 DIM PHRASE$(100)
105 DATA PIERRE DUPONT,7 RUE RENOIR,9742
9 VILLAGE
110 DATA VEUILLEZ CROIRE A L'EXPRESSION
DE MES SENTIMENTS LES MEILLEURS,"
      SIGNATURE"
120 CALL CHAR(120,"813C7EFFFF7E3C81")
130 CALL COLOR(12,9,2)
140 CALL CLEAR
150 X=3
160 Y=4
170 PH=0
180 CALL SOUND(50,400,2)
190 CALL KEY(0,C,S)
200 CALL GCHAR(Y,X,CAR)
210 CALL HCHAR(Y,X,120)
220 CALL HCHAR(Y,X,CAR)
230 IF S=0 THEN 190
240 IF C=6 THEN 360
250 IF C=8 THEN 590
260 IF C<>15 THEN 290
270 READ TAMPON$
280 GOTO 690
290 IF C=13 THEN 690
300 TAMPON$=TAMPON$&CHR$(C)
310 GOSUB 520
320 GOTO 180
330 REM ===================
340 REM REPRODUIRE LE TEXTE
350 REM ===================
360 CALL CLEAR
370 X=3
380 Y=4
390 FOR I=1 TO PH
400 R$=PHRASE$(I)
410 FOR J=1 TO LEN(R$)
420 C=ASC(SEG$(R$,J,1))
430 GOSUB 520
440 NEXT J
450 Y=Y-(Y<24)
460 X=1
470 NEXT I
480 GOTO 180
490 REM ===================
500 REM LE CURSEUR AVANCE
510 REM ===================
520 CALL HCHAR(Y,X,C)
530 X=-(X+1)*(X<32)-(X=32)
540 Y=Y-(X=1)
550 RETURN
560 REM ===================
570 REM LE CURSEUR RECULE
580 REM ===================
590 T=LEN(TAMPON$)
600 X=((X-1)*(X>1)+32*(X=1))*(T<>1)-X*(T
=1)
610 Y=-(T<>1)*(Y+(X=32))-Y*(T=1)
620 CALL HCHAR(Y,X,32)
630 REM
640 TAMPON$=SEG$(TAMPON$,1,T+(T>0))
650 GOTO 180
660 REM ===================
670 REM PHRASE ENREGISTREE
680 REM ===================
690 PH=PH+1
700 PHRASE$(PH)=TAMPON$
710 TAMPON$=""
720 X=1
730 Y=Y-(Y<24)
740 GOTO 180
```

Introduction au Pascal

Gérard Santraille

On parle beaucoup de Pascal : on prétend qu'il pourrait supplanter de nombreux autres langages dans des domaines aussi variés que le calcul scientifique, les mathématiques, la gestion, les programmes éducatifs (désormais appelés "didacticiels") et même les jeux.

Pascal serait donc LE langage universel, parfaitement polyvalent et facile d'emploi. Qu'en est-il réellement ? Même si de nos jours Pascal est assez largement diffusé (notamment dans les milieux universitaires), Fortran, Cobol et le Basic ont encore de beaux jours devant eux.

Une anecdote croustillante : de mauvaises langues assurent que la sonde américaine Mariner IV serait encore de ce monde si son programme de contrôle de trajectoire avait été écrit en Pascal. En effet, la commission chargée d'étudier les causes de l'accident révéla que l'erreur était imputable à un programme Fortran qui avait confondu une variable avec un mot-clé suite à une faute de frappe. Cette mésaventure ne pourrait se produire en Pascal où chaque variable est déclarée.

Les scientifiques n'ont pas pour autant abandonné Fortran. Les programmes Pascal sont cependant de plus en plus nombreux sur le marché industriel (on y trouve même des "simulateurs de procédés", domaine jusqu'alors réservé au Fortran).

L'apparition du Pascal UCSD (facilement implantable sur les micro-ordinateurs comme le TI-99 ou l'Apple) a généralisé ce processus et, désormais, chacun peut se livrer aux joies du Pascal avec une configuration somme toute réduite et peu coûteuse par rapport au matériel nécessaire il y a seulement quelques années.

Pascal est un langage de haut niveau (tout comme le BASIC...), particulièrement bien adapté à la programmation structurée. L'écriture d'un programme en Pascal impose une certaine méthode d'analyse. L'apprentissage d'un tel langage passe par la maîtrise parfaite de cette dernière souvent baptisée "analyse descendante".

Cette formule recouvre un processus de décomposition (découpage par blocs) qu'il conviendra d'appliquer parfaitement si l'on ne veut pas perdre trop de temps lors de la mise au point des programmes. Ce type de découpage permet d'affiner progressivement la résolution du problème. Il s'avère particulièrement utile lorsqu'on ne peut embrasser mentalement l'ensemble des étapes requises pour un traitement.

Le Pascal UCSD et le TI-99

Pour pratiquer Pascal sur notre ordinateur préféré, il faut d'abord disposer de la carte P-Code, chargée d'interpréter les programmes écrits en ce langage. Sont également nécessaires l'extension 32 K et un lecteur de disquette au moins (avec contrôleur bien entendu).

Si l'on désire développer soi-même des programmes, il faut se munir de la disquette Editeur/Filer/Utilitaires (pour les écrire, les sauvegarder...) et de la disquette Compilateur (pour les compiler et les exécuter). Enfin, pour utiliser des programmes en langage assembleur (TMS 9900) sous système-P, il faut disposer de la disquette Assembleur/Editeur de liens.

La disquette Editeur/Filer/Utilitaires comporte un éditeur de texte pour l'écriture et la mise au point des programmes, un système d'exploitation de disques appelé Filer (qui permet de dupliquer, sauvegarder et manipuler les différents fichiers) et un ensemble de programmes utilitaires. Quant à la disquette Compilateur, elle transforme un fichier TEXT (issu de l'éditeur) en un fichier CODE exécutable par l'ordinateur.

Il faut savoir que "Pascal UCSD" est une marque déposée de "l'Université de Californie de San-Diego" et que la "portabilité" d'un logiciel écrit dans ce langage est théoriquement totale, c'est-à-dire que si votre grand-oncle a mis au point un fantastique programme de calcul de probabilités au loto sur un Apple II ou sur un TRS 80, vous pourrez théoriquement l'utiliser tel quel sur votre TI avec la carte d'interprétation du code-P (la réciproque se vérifie également).

La documentation fournie par Texas Instruments (en anglais) suppose que l'on sait déjà programmer en Pascal standard. Elle ne présente que les caractéristiques particulières du Pascal UCSD, notamment de la version IV.0 qui est celle implantée sur le TI-99.

Les ouvrages d'introduction au Pascal (standard ou UCSD) sont nombreux. Une bibliographie sommaire vous est d'ailleurs donnée à la fin de cet article.

Si, comme je l'espère, vous êtes un passionné de programmation (le simple fait de lire cette revue tendrait à le prouver), il vous faut connaître et pratiquer le Pascal. Ce langage d'une richesse inouïe, tant par ses capacités graphiques que par ses aptitudes au calcul ou à la gestion, va progressivement s'implanter dans la plupart des domaines de l'informatique professionnelle ou personnelle. Ignorer totalement Pascal pourrait un jour constituer un handicap.

Bibliographie sommaire

Initiation

- Le Pascal par l'exemple, de Jean-Alain Hernandez, aux Editions Eyrolles.
- Du Basic au Pascal, de E. Floegel, aux Editions Micro Systèmes.
- Introduction au P-System UCSD, de J. Butah et C. Grant, aux Editions SYBEX.
- Le système Pascal UCSD (2 tomes), de Thierry Chamoret, aux Editions Editests.
- Programmer en Pascal, de Daniel-Jean David et Jean-Luc Deschamps, aux Editions du PSI.

Perfectionnement

- Le langage Pascal 150 avec Pascal 6000 et Pascal UCSD, de Nadia Magnetat-Thalmann, Daniel Thalmann et Jean Vaucher, aux Editions Gaétan Morin.
- Pascal (manuel de base, techniques et concepts de programmation avancés), de P. Lignelet, aux Editions EAP.

Applications

- Programmation structurée en langage Pascal, de R. Lorthal, aux Editions Masson.
- Programmes en Pascal pour scientifiques et ingénieurs, de Alan R. Miller, aux Editions Sybex.

Puissance 4

Daniel Massé

Il y a quelques années déjà, je passais bien des heures à jouer au "morpion", pas tellement pendant les heures de permanence (il y avait toujours autre chose à faire), mais surtout pendant les heures un peu embêtantes (comme par exemple en anglais, et surtout en histoire, qui me "barbait" profondément).

Ce temps est maintenant bien passé, et je manquais un peu de partenaires, jusqu'à ce que je trouve cet adversaire implacable qu'est le TI-99/4A.

Le jeu de Puissance 4 est assez semblable au Morpion, avec la seule différence que le jeu se joue normalement en glissant des jetons dans une grille verticale : les jetons tombent en bas de la colonne dans laquelle on les place. Le gagnant est celui qui aligne 4 pions, dans n'importe qu'elle direction.

Voyons maintenant un peu le programme. Après avoir défini plusieurs valeurs de N (sur lesquelles nous reviendrons plus loin), nous définissons les caractères représentant les pions des deux joueurs, puis une petite musique qui rend le programme plus vivant. Les lignes 410 à 450 permettent au joueur de choisir le niveau de difficulté ND (égal à 0 ou à 1), dont nous verrons plus loin comment il affecte la tactique de l'ordinateur.

Les instructions suivantes (jusqu'à 720) décrivent le coup du joueur, et sont simples à comprendre. Au passage, nous voyons les variables JC et SC, définissant les couleurs d'écran et de caractères, la variable C qui accumule le nombre de coups joués depuis le début de la partie, G(X) qui représente le nombre de pions déjà positionnés dans la colonne X, et la variable A(Y,X) qui met en mémoire le pion joué dans la case (Y,X). A est égal à 1 si le pion appartient au joueur, et à 5 si le pion appartient à l'ordinateur.

La partie la plus intéressante du programme commence en 720 : c'est la tactique de l'ordinateur. A partir de la ligne 790, et pour chaque valeur de X, l'ordinateur commence par regarder s'il reste une case libre dans la colonne, puis va consulter les cases voisines pour voir quels sont les pions déjà alignés avec la case libre.

Il commence par regarder les cases se trouvant dans la même colonne (alignements verticaux), puis celles qui se trouvent dans la même ligne (alignements horizontaux), et finalement les diagonales. C'est dans l'évaluation de ces alignements qu'intervient le tableau N défini au début du programme, et que nous retrouvons en 1800.

La variable S additionne les valeurs de A(Y,X) pour chaque case de l'alignement. Par exemple, si l'alignement contient déjà 3 pions de l'adversaire (c'est-à-dire vous), A prend la valeur 1 pour chacun de ces pions, et S a donc la valeur 3. Il est évident que l'ordinateur, s'il joue correctement, doit jouer sur cette case (à moins qu'il puisse gagner en alignant un quatrième pion avec 3 autres déjà en place). La variable N(S+1), égale à N(4), est égale à 1000.

L'ordinateur met en mémoire une variable T représentant la somme des valeurs de N pour chacun des alignements passant par la case considérée. Pour jouer, il choisira la case ayant permis d'obtenir la plus grande valeur de T.

Mais vous pourrez vérifier vous-même que vous battez très facilement l'ordinateur quand vous choisissez le niveau 0, car l'ordinateur ne s'occupe pas de savoir si, en jouant, il ne vous donne pas un marche-pied permettant de gagner au coup suivant.

C'est pourquoi, au niveau 1 (ND=1), l'ordinateur va regarder ce qui se passerait au coup suivant : il évalue les alignements passant par la case située au-dessus de celle où il envisage de jouer, pour éviter autant que possible que vous puissiez gagner au coup suivant. Vous constaterez alors qu'il devient beaucoup plus difficile à battre. En fait, il serait possible de le rendre pratiquement imbattable, en affinant encore sa tactique, mais il faut reconnaître qu'il n'est pas très drôle de se faire toujours battre par une vulgaire machine... Et vous verrez vous-même qu'il faut déjà un certain temps pour trouver une tactique victorieuse (ce qui, bien entendu, est tout à fait possible).

Ecrit en **Basic Etendu**, ce programme n'utilise cependant pas toutes les possibilités de ce langage, en dehors de quelques instructions multiples et de l'instruction DISPLAY AT, bien préférable au PRINT (qui entraine un "SCROLL" désagréable). Avec quelques modifications, vous devez pouvoir réécrire le programme en Basic TI, mais il sera alors nettement plus lent.

L'instruction de la ligne 280 (ON ERROR 1880) a été laissée à titre de démonstration : elle est bien utile pour mettre au point un programme. L'instruction de la ligne 160 (ON WARNING NEXT) est importante quand vous rentrez des données par l'instruction ACCEPT : si vous utilisez VALIDATE et que vous tapez ENTER, vous obtenez un message d'erreur. Si vous n'utilisez pas VALIDATE et tapez autre chose que le chiffre attendu, vous obtenez aussi un message d'erreur.

```
100 ! ******************
105 ! *  PUISSANCE  4  *
110 ! ******************
115 ! *  BASIC ETENDU  *
120 ! ******************
125 ! *   COPYRIGHT    *
130 ! *  DANIEL MASSE  *
140 ! * ET 99 MAGAZINE *
145 ! ******************
```



```
146 !
148 CALL CLEAR
150 DIM A(5,7),N(20)
160 ON WARNING NEXT
170 N(1)=.01
180 N(2)=.1
190 N(3)=10
200 N(4)=1000
210 N(6)=1
```

```
220 N(11)=100
230 N(16)=10000
240 CALL CHAR(104,"0101030307OF3EFC8080C
0C0E0F07C3FFC3E0F07030301013F7CF0E0C0C08
080")
250 CALL COLOR(10,13,4)
260 CALL CHAR(112,"804020181C0E070301020
4183870E0C003070E1C18204080C0E0703818040
201")
270 CALL COLOR(11,5,12)
280 ON ERROR 1880
290 JC=14 :: SC=2 :: K=1
300 GOSUB 2270
310 DISPLAY AT(1,10)ERASE ALL:"PUISSANCE
 4" :: GOSUB 320 :: GOTO 410
320 CALL SOUND(100,494,2)
330 CALL SOUND(100,440,2)
340 CALL SOUND(400,494,2)
350 CALL SOUND(100,440,2)
360 CALL SOUND(100,392,2)
370 CALL SOUND(100,349,2)
380 CALL SOUND(200,330,2)
390 CALL SOUND(400,294,2)
400 RETURN
410 DISPLAY AT(2,10):"==========="
420 DISPLAY AT(10,5):"NIVEAU DE DIFFICUL
TE ?"
430 DISPLAY AT(12,5):"O POUR DEBUTANT","
 ":"     1 POUR JOUEUR CONFIRME"
440 ACCEPT AT(10,27):ND
450 IF ND<>0 AND ND<>1 THEN 440
460 REM JEU
470 GOSUB 1910
480 DISPLAY AT(3,0):"OBJECTIF : ALIGNER
4 PIONS","LA DIRECTION EST QUELCONQUE"
490 DISPLAY AT(6,0):"TIRAGE AU SORT DU 1
ER JOUEUR"
500 FOR I=1 TO 1000 :: NEXT I
510 RANDOMIZE
520 JO=INT(RND*2)+1
530 IF JO=1 THEN 720
540 REM JOUEUR JOUE
550 JC=13 :: SC=4 :: GOSUB 2270
560 C=C+1
570 IF C=36 THEN 1510
580 DISPLAY AT(3,0):"A VOUS DE JOUER","N
O DE COLONNE S.V.P. = "
590 DISPLAY AT(6,0):" "
600 ACCEPT AT(4,24)BEEP:X
610 IF X<1 OR X>7 THEN 600
620 IF G(X)<5 THEN 650
630 DISPLAY AT(6,0):"COLONNE REMPLIE. RE
JOUEZ"
640 FOR I=1 TO 200 :: NEXT I :: DISPLAY
AT(6,0):" " :: GOTO 600
650 Y=5-G(X)
660 CAR=104
670 G(X)=G(X)+1
680 A(Y,X)=1
690 GOSUB 1450
700 IF L(X)=1 THEN 1570
710 GOTO 720
720 REM TI JOUE
730 C=C+1
740 B=0
750 M=-10000
760 IF C=36 THEN 1510
770 GOSUB 1820
780 FOR X=1 TO 7
790 TT=0 :: GOSUB 860
800 IF ND THEN GOSUB 1400
810 IF T<M THEN 840
820 M=T
830 XM=X
840 NEXT X
850 GOTO 1290
860 L(X)=0
870 T=0
880 Y=5-G(X)
890 IF (Y=0 AND TT=0)OR(Y<0 AND TT=1)THE
N T=-100000 :: RETURN
900 IF Y=5 THEN 950
910 S=0
920 FOR J=1 TO 4 :: S=S+A(J,X):: NEXT J
930 GOSUB 1740
940 IF Y=1 THEN 970
950 S=0
960 FOR J=2 TO 5 :: S=S+A(J,X):: NEXT J
970 GOSUB 1740
980 IF X>4 THEN 1050
990 FOR D=1 TO X
1000 S=0
1010 FOR I=D TO D+3 :: S=S+A(Y,I):: NEXT
 I
1020 GOSUB 1740
1030 NEXT D
1040 GOTO 1100
1050 FOR D=X-3 TO 4
1060 S=0
1070 FOR I=D TO D+3 :: S=S+A(Y,I):: NEXT
 I
1080 GOSUB 1740
1090 NEXT D
1100 AA=X+Y
1110 IF Y=5 OR AA>8 OR AA<5 THEN 1150
1120 S=0
1130 FOR J=1 TO 4 :: I=AA-J :: S=S+A(J,I
):: NEXT J
1140 GOSUB 1740
1150 IF Y=1 OR AA>9 OR AA<6 THEN 1190
1160 S=0
1170 FOR J=2 TO 5 :: I=AA-J :: S=S+A(J,I
):: NEXT J
1180 GOSUB 1740
1190 AA=X-Y
1200 IF Y=5 OR AA<0 OR AA>3 THEN 1240
1210 S=0
1220 FOR J=1 TO 4 :: I=AA+J :: S=S+A(J,I
```

```
):: NEXT J
1230 GOSUB 1740
1240 IF Y=1 OR AA<-1 OR AA>2 THEN 1280
1250 S=0
1260 FOR J=2 TO 5 :: I=AA+J :: S=S+A(J,I
):: NEXT J
1270 GOSUB 1740
1280 RETURN
1290 X=XM
1300 DISPLAY AT(4,12):X
1310 FOR I=1 TO 200 :: NEXT I
1320 CAR=112
1330 Y=5-G(X)
1340 G(X)=G(X)+1
1350 A(Y,X)=5
1360 GOSUB 1450
1370 B=1 :: TT=0
1380 GOSUB 860
1390 GOTO 540
1400 REM EVALUE CASE DESSUS
1410 Y=Y-1 :: TT=1
1420 GOSUB 890
1430 Y=Y+1
1440 RETURN
1450 REM AFFICHE PION
1460 CALL HCHAR(7+3*Y,3*X,CAR)
1470 CALL HCHAR(7+3*Y,1+3*X,CAR+1)
1480 CALL HCHAR(8+3*Y,3*X,CAR+2)
1490 CALL HCHAR(8+3*Y,1+3*X,CAR+3)
1500 RETURN
1510 REM EGALITE
1520 JC=14 :: SC=2 :: GOSUB 2270
1530 DISPLAY AT(3,0):"EGALITE. PAS MAL J
OUE"," ":" ":" " :: GOSUB 320
1540 FOR I=1 TO 1000 :: NEXT I
1550 GOSUB 1910
1560 GOTO 490
1570 REM JOUEUR GAGNE
1580 JC=14 :: SC=2 :: GOSUB 2270
1590 DISPLAY AT(3,0):"OK VOUS AVEZ GAGNE
"," ":" ":" " :: GOSUB 320
1600 R(1)=R(1)+1
1610 FOR I=1 TO 1000 :: NEXT I
1620 GOSUB 1910
1630 GOTO 720
1640 REM TI GAGNE
1650 DISPLAY AT(3,0):"DESOLE, JE JOUE EN
 ";X
1660 CAR=112
1670 GOSUB 1450
1680 JC=14 :: SC=2 :: GOSUB 2270
1690 DISPLAY AT(4,0):"ET JE GAGNE" :: GO
SUB 320
1700 FOR I=1 TO 1000 :: NEXT I
1710 R(2)=R(2)+1
1720 GOSUB 1910
1730 GOTO 540
1740 REM EVALUATION
1750 IF S=3 AND TT=0 THEN L(X)=1
1760 IF S=3 AND TT=1 THEN T=T-1000
1770 IF S=2 AND TT=1 THEN T=T-10
1780 IF TT=1 THEN 1810
1790 IF S=15 AND B=0 THEN 1640
1800 T=T+N(S+1)
1810 RETURN
1820 REM ECRITURE
1830 JC=5 :: SC=8 :: GOSUB 2270
1840 DISPLAY AT(3,0):"A MOI DE JOUER","J
E JOUE EN"
1850 DISPLAY AT(6,0):" "
1860 RETURN
1870 GOTO 1800
1880 CALL ERR(CD,TY,SE,LN)
1890 PRINT CD;TY;SE;LN
1900 RETURN NEXT
1910 REM TRACE DE LA GRILLE
1920 JC=9 :: SC=15
1930 K=1
1940 GOSUB 2270
1950 DISPLAY AT(1,10)ERASE ALL:"PUISSANC
E 4"
1960 DISPLAY AT(2,10):"==========="
1970 CALL CHAR(96,"000000FF")
1980 CALL CHAR(97,"3C3C3C3C3C3C3C3C")
1990 CALL CHAR(98,"FFFFFFFFFFFFFFFF")
2000 FOR I=9 TO 24 STEP 15
2010 CALL HCHAR(I,2,98,22)
2020 NEXT I
2030 FOR I=12 TO 21 STEP 3
2040 CALL HCHAR(I,2,96,22)
2050 NEXT I
2060 FOR I=2 TO 23 STEP 3
2070 CALL VCHAR(10,I,97,14)
2080 NEXT I
2090 FOR I=1 TO 19 STEP 3
2100 DISPLAY AT(8,I):(I+2)/3
2110 NEXT I
2120 FOR J=1 TO 7
2130 L(J),G(J)=0
2140 FOR I=1 TO 5
2150 A(I,J)=0
2160 NEXT I
2170 NEXT J
2180 C=0
2190 DISPLAY AT(9,23):"SCORE"
2200 DISPLAY AT(8,23):"====="
2210 DISPLAY AT(10,23):"====="
2220 DISPLAY AT(12,22):"JOUEUR"
2230 DISPLAY AT(14,24):R(1)
2240 DISPLAY AT(17,23):"T.I."
2250 DISPLAY AT(19,24):R(2)
2260 RETURN
2270 REM COULEUR ECRITURE
2280 FOR I=0 TO 8
2290 CALL COLOR(I,JC,K)
2300 CALL SCREEN(SC)
2310 NEXT I
2320 RETURN
```

Les modes d'adressage en assembleur

Jean-Luc Bazanegue

Cet article est destiné aux lecteurs qui possèdent depuis peu l'Editeur/Assembleur (ou la mini-mémoire). Il est évident que ceux qui pratiquent l'assembleur avec maestria n'apprendront pas grand chose en lisant ces pages.

Pour ce premier article sur l'assembleur, il nous a semblé important d'écrire quelques lignes sur les différents modes d'adressage du microprocesseur TMS 9900. Une bonne maîtrise de ces techniques est en effet indispensable lorsque l'on veut aborder ce langage de programmation. Avant d'analyser chaque mode d'adressage, et afin de ne pas mettre la charrue avant les boeufs, quelques généralités s'imposent.

Notre TI-99/4A, comme tout micro-ordinateur normalement constitué, est composé d'un certain nombre d'éléments parmi lesquels nous trouvons la mémoire, elle même divisée en deux grandes familles :

- la *mémoire vive* (en anglais Random Acces Memory) dans laquelle seront stockées les données et instructions constituant un programme ;
- la *mémoire morte* (en anglais Read Only Memory) qui contient différents programmes figés. L'interpréteur BASIC est un de ces programmes inaltérables, accessibles dès la mise sous tension du TI-99/4A.

Ces deux types de mémoire constituent la *mémoire adressable*. Le terme "adressable" s'explique lorsque l'on sait que la mémoire est divisée en plusieurs milliers de "cases" numérotées.

Le microprocesseur va nous permettre de gérer ces adresses. Le TI-99/4A a été construit autour du microprocesseur TMS 9900 qui est le "cerveau" de notre ordinateur ("cerveau" est un bien grand mot puisque l'on sait que le plus puissant des ordinateurs est à peu près aussi intelligent qu'un fer à repasser !). Le traitement des données ou informations situées en mémoire vive sera réalisé par le microprocesseur. Pour rendre ce travail possible, le programmeur devra lui indiquer l'adresse en mémoire de l'information à traiter et la nature du traitement à appliquer. Dans ce but, les créateurs de notre TMS 9900 préféré ont prévu un jeu d'instructions qui constitue le "langage" assembleur.

Qu'est ce que l'assembleur ?

Le microprocesseur ne reconnaît que les codes binaires (base 2). La manipulation de ces codes n'est pas aisée et un mode de communication plus humain s'est rapidement imposé. Ce système, c'est l'assembleur.

L'assembleur est un langage à base de *codes mnémoniques* (seulement pour les anglophones, hélas !) ; l'assembleur est aussi un traducteur : il traduit ces codes binaires directement exécutables par le microprocesseur. Les codes mnémoniques constituent le *code source* et les codes binaires (ou machine) générés par l'assembleur constituent le *code objet.*

Base 10 et base 16

L'assembleur reconnaît deux systèmes de numération :

- Le système décimal (base 10) est utilisé par défaut. Nous le connaissons bien puisque nous l'employons tous depuis l'école maternelle ;
- Le système hexadécimal (base 16), utilisé en TI Basic pour redéfinir les caractères ASCII (CALL CHAR), permet une représentation concise des nombres binaires. Pour que l'assembleur puisse faire la différence avec les nombres décimaux, les nombres hexadécimaux doivent être précédés du signe "plus grand que" (>). Ainsi, ">A9" correspond à "169" en décimal.

Octets et mots

Nous avons vu précédemment que la mémoire est constituée de plusieurs "cases". Chacune de ces cases est un octet (Byte en anglais). Chaque octet est composé de 8 éléments binaires (BIT pour BInary digiT en anglais). Le microprocesseur TMS 9900 peut adresser 65536 octets numérotés de 0 à 65535 (>0000 à >FFFF en hexadécimal). Le TMS 9900 étant un microprocesseur 16 bits, il peut traiter simultanément deux octets. Dans ce cas, les deux octets ainsi adressés constituent un mot mémoire (Word en anglais).

La mémoire de travail

La *mémoire de travail,* aussi appelée zone de travail, est un groupe de 32 octets soit 16 mots. Ces mots sont des registres de mémoire utilisés fréquemment par le programmeur pour des stockages temporaires et par le microprocesseur pour l'exécution de certaines instructions. Ces registres n'ayant pas d'adresses absolues, le programmeur doit indiquer l'adresse du premier d'entre eux.

Les directives d'assembleur

Ce sont des commandes destinées à faciliter le travail du programmeur. Elles permettent de donner des noms aux adresses mémoire, définir la zone de mémoire où se trouvera le programme, réserver des blocs de mémoire (tampon), etc...

L'instruction MOVB

Pour illustrer les différents modes d'adressage, il est pratique de prendre l'exemple d'une instruction (code mnémonique). J'ai choisi l'instruction MOVB (MOVe Byte) pour sa simplicité. Cette instruction réclame deux adresses, l'adresse source et l'adresse destination. MOVB copie la valeur contenue dans l'adresse source (octet) vers l'adresse de destination. L'adresse source n'est pas modifiée. Cette instruction ne traite que des octets mais une autre similaire (MOV) est prévue pour traiter des mots. Les deux adresses doivent être séparées par une virgule.

Adressage des registres de mémoire de travail

L'adresse du premier registre de la mémoire de travail doit être placée dans un des registres spéciaux intégrés au microprocesseur. Celui-ci est baptisé "registre pointeur de mémoire de travail". L'instruction LWPI (Load Workspace Pointer Immediate) exécute cette opération. Par exemple :

LWPI >A000

charge la valeur >A000 dans le registre pointeur. Les adresses des 16 registres de mémoire de travail seront calculées automatiquement d'après cette valeur. Une commande de l'assembleur permet d'affecter à ces registres les symboles R0 à R15. Notons pour mémoire que, par convention, on appelle A0 le *poids fort* de l'adresse A000, et 00 le *poids faible.* Ainsi, de manière générale, le poids fort d'une adresse WXYZ est contitué par WX (octet de gauche) et son poids faible par YZ (octet de droite).

TRES IMPORTANT : un registre de mémoire de travail est constitué de deux octets (16 bits). Si nous employons une instruction qui ne traite qu'un octet (comme MOVB), l'octet adressé sera toujours l'octet de poids

fort alors que l'octet de poids faible restera inchangé.

Un petit exemple :

Le registre R0 contient la valeur >0000 et le registre R1 contient la valeur >12AA. Exécutons l'instruction suivante :

MOVB R0,R1

Le registre R0 n'est pas modifié et le registre R1 contient maintenant la valeur >00AA. l'octet de poids faible (octet de droite) du registre R1 n'a pas changé.

Adressage indirect par l'intermédiaire d'un registre de mémoire de travail

Avec le mode indirect, il est possible d'adresser un octet ou un mot en fonction de la valeur contenue dans un des registres de mémoire de travail. Dans ce cas, celui ci doit être précédé d'un astérisque (*).

Dans l'exemple suivant, l'octet source est adressé par le mode registre de mémoire de travail et l'octet destination par le mode indirect. Il est donc possible de combiner deux modes d'adressage différents.

Exemple :

- Le registre R0 contient la valeur >FF00
- Le registre R1 contient >AA88
- L'octet situé à l'adresse >AA88 contient >1F

Si l'on exécute l'instruction :

MOVB R0,*R1

une copie du contenu de l'octet de poids fort du registre R0 est reportée dans l'octet dont l'adresse est située dans le registre R1.

Les registres R0 et R1 n'ont pas changé alors que l'octet situé à l'adresse >AA88 contient maintenant la valeur >FF au lieu de >1F.

Adressage indirect par l'intermédiaire d'un registre de mémoire de travail auto-incrémenté

Ce mode d'adressage est similaire à celui évoqué ci-dessus avec en plus une auto-incrémentation du registre. Après exécution, le microprocesseur ajoute automatiquement 1 au registre si l'instruction traite un octet ou 2 si elle traite un mot.

Ceci est très pratique lorsqu'il faut manipuler plusieurs octets ou mots dont les adresses se suivent.

Le registre doit être précédé d'un astérisque (*) et suivi du signe "plus" (+).

Exemple :

- Le registre R5 contient la valeur >1000
- Le registre R8 contient >B000
- L'octet situé à l'adresse >B000 contient >00

Si l'on exécute l'instruction :

MOVB R5,*R8+

une copie de l'octet de poids fort du registre R5 est reportée dans l'octet dont l'adresse se trouve dans le registre R8. L'octet situé à l'adresse >B000 contient la valeur >10 au lieu de >00. Le registre R8 contient maintenant >B001 et "pointe" ainsi sur l'octet suivant.

Adressage symbolique (direct)

L'assembleur offre la possibilité d'affecter des symboles aux adresses. Un symbole peut contenir jusqu'à 6 caractères (2 pour l'assembleur ligne par ligne de la mini-mémoire) et doit être précédé du signe "arrobas" (@). Le premier caractère doit avoir un code ASCII compris entre 65 et 90 (une lettre majuscule). Les caractères suivants peuvent être des lettres majuscules ou des chiffres. Les directives (commandes) d'assembleur BSS, BES, PEND, EQU, BYTE, DATA et TEXT permettent l'affectation de symboles aux adresses.

NOTE : Les directives BES, PEND et BYTE ne sont pas utilisables avec l'assembleur ligne par ligne de la mini-mémoire.

Exemple :

- Le symbole SYMB1 est affecté à l'adresse >F000
- Le symbole SYMB2 est affecté à l'adresse >FF00
- L'octet situé à l'adresse >F000 contient la valeur >7C
- L'octet situé à l'adresse >FF00 contient >1A

Si l'on exécute l'instruction :

MOVB @SYMB1,@SYMB2

une copie du contenu de l'octet situé à l'adresse >F000 est reportée dans l'octet situé à l'adresse >FF00. L'octet situé à l'adresse symbolique SYMB2 contient maintenant la valeur >7C alors que celui situé à l'adresse SYMB1 n'a pas changé.

Notons que nous aurions obtenu les mêmes résultats avec l'instruction :

MOVB @>F000,@>FF00

Adressage symbolique indexé

L'adresse n'est plus obtenue directement comme ci-dessus mais en ajoutant à l'adresse symbolique la valeur contenue dans un pointeur (index). Un registre de mémoire de travail est utilisé comme pointeur.

TRES IMPORTANT : Le registre de mémoire de travail R0 ne peut pas être utilisé comme pointeur.

Le registre doit être placé entre parenthèses à la suite de l'adresse symbolique.

Exemple :

- Le registre R5 contient la valeur >AA00
- Le registre R1 contient >0008

Exécutons l'instruction :

MOVB R5,@>B000(R1)

Une copie du contenu de l'octet de poids fort du registre R5 est reportée dans l'adresse calculée en ajoutant la valeur contenue dans R1 à >B000. L'octet situé à l'adresse >B008 contient maintenant la valeur >AA.

En supposant que le registre R1 ait été égal à zéro, l'instruction :

MOVB R5,@>B000

aurait été équivalente à l'instruction précédente puisque >B000 + >0000 = >B000.

Adressage immédiat

Les instructions qui utilisent ce mode d'adressage ne traitent que des mots et se terminent toujours par la lettre I. LWPI, rencontrée précédemment, fait partie de cette famille. Le mot adressé est situé immédiatement après le code mnémonique.

L'instruction LI (Load Immediate) charge immédiatement une valeur dans un registre de mémoire de travail. Par exemple, l'instruction :

LI R0,>1800

charge immédiatement la valeur >1800 dans le registre R0.

Autres modes d'adressage

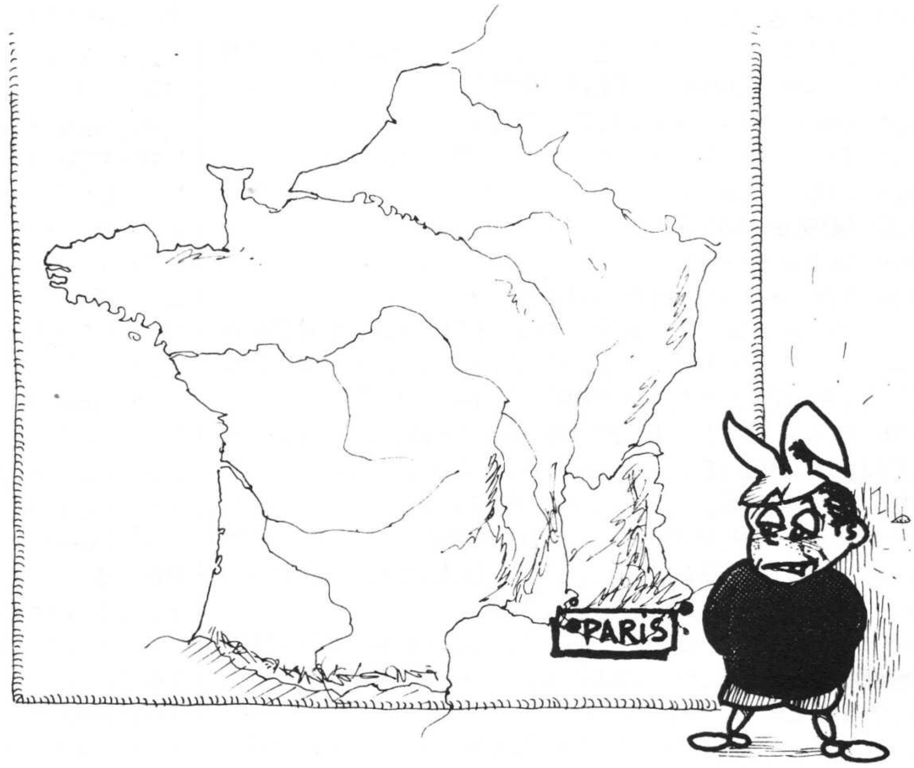
Avant de conclure cet article, il faut signaler l'existence de deux autres modes d'adressage :

- l'adressage relatif, employé pour les sauts conditionnels (tests) ;
- l'adressage de l'unité de communications, employé pour les entrées/sorties (unités de disquettes, imprimantes, etc...).

Nous ne nous étendrons pas sur ces modes d'adressage qui sont sensiblement plus complexes que ceux analysés dans cette première initiation. Nous aurons l'occasion d'en parler dans un prochain article.

Géographie

Laurent Desobelle



NDLR : *Ce programme est une contribution de Monsieur Laurent Desobelle, un de nos lecteurs assidus, et fait suite à notre petit jeu "A vos claviers". Comme promis, Laurent reçoit un abonnement d'un an avec les cassettes d'accompagnement.*

Nous avons choisi ce programme, écrit en **Basic TI**, pour son graphisme très soigné. Nous imaginons facilement les nombreuses heures qui ont été nécessaires pour la représentation de la carte de France.

Ce programme comportait, lorsque nous l'avons reçu, quelques défauts, et nous nous sommes permis d'y apporter les modifications nécessaires. Dans un but pédagogique, nous vous livrons la version que nous a envoyé notre ami ainsi que la version corrigée. Vous pourrez ainsi comparer les deux programmes et peut-être éviter certaines erreurs.

Modifications apportées au programme

Couleurs : les caractères étaient en blanc sur gris et nous avons préféré les faire passer en vert foncé sur jaune clair afin d'obtenir un meilleur contraste.

Protection des saisies : au moment de la conception d'un programme, il faut toujours penser qu'il pourrait être utilisé par quelqu'un d'autre ou, qui plus est, par une personne qui touche à un ordinateur pour la première fois. Il est donc nécessaire de guider l'utilisateur et d'éviter un éventuel "plantage" du programme par pression d'une touche "inadéquate". C'est pourquoi nous avons modifié le segment de programme chargé du choix du niveau de jeu.

Tirage aléatoire des villes : dans la version de base, une ligne de DATAs est choisie de façon aléatoire et les villes apparaissent ensuite séquentiellement, ce qui peut donner au programme un aspect répétitif. Dans la version modifiée, les villes sont choisies par une boucle FOR-NEXT dans laquelle la variable d'arrivée contient une valeur aléatoire.

Fin de programme : pour sortir du programme et ainsi avoir le résultat du jeu (note), il fallait poursuivre jusqu'à la fin de la liste des DATAs. Il est maintenant possible d'arrêter le programme en appuyant sur la touche "égal" (=).

Suppression des ELSE inutiles : en Basic TI, le ELSE est rarement utile après un IF-THEN. Nous les avons donc supprimés car il encombraient la mémoire inutilement et ralentissaient le programme.

Suppressions des "0" à la fin des DATAs destinés aux instructions CALL CHAR : lors de la redéfinition d'un caractère par l'instruction CALL CHAR, les "0" en fin de redéfinition sont inutiles car rajoutés automatiquement par défaut.

Ces modifications permettent d'améliorer un programme déjà très intéressant. Si, comme moi, vous n'êtes pas particulièrement fort en géographie, ce programme vous permettra d'améliorer sensiblement vos connaissances.

```
100 REM ****************
110 REM *  GEOGRAPHIE  *
120 REM ****************
130 REM *   TI BASIC   *
140 REM ****************
150 REM *  COPYRIGHT   *
160 REM *   LAURENT    *
170 REM *  DESOBELLE   *
180 REM *      ET      *
190 REM * 99  MAGAZINE *
200 REM ****************
210 REM ** DEBUT PARTIE **
220 CALL CLEAR
230 RANDOMIZE
240 CALL CHAR(158,"1898FF3D3C3CE404")
250 CALL CHAR(159,"1819FFBC3C3C272")
260 DIM CA(18,11)
270 CALL SCREEN(12)
280 FOR Q=1 TO 16
290 CALL COLOR(Q,13,12)
300 NEXT Q
310 Q=0
320 W$="CE PROGRAMME PERMET   DE TESTER
VOS          CONNAISSANCES EN       GEOGRA
```

```
PHIE"
330 P=2
340 A=15
350 GOSUB 790
360 F=0
370 DIM D(2)
380 DIM S(2)
390 PRINT "CHOISISSEZ VOTRE FORCE":"1=FO
RT":"2=MOYEN":"3=DEBUTANT"
400 CALL KEY(0,KEY,STATUS)
410 IF (KEY<49)+(KEY>51)THEN 400
420 CALL CLEAR
430 GOSUB 1420
440 SCD=KEY-48
450 REM * FICHIER VILLES *
460 DATA CALAIS,3,23,MONTPELLIER,18,25,A
NNECY,13,27,BOULOGNE-S-M,4,22,TOULON,18,
27,BIARRITZ,17,19,NANTES,10,19
470 DATA BREST,8,17,VERSAILLES,7,23,CHAR
LEVILLE-MEZIERES,5,25,AIX-EN-PROVENCE,18
,26,LE HAVRE,5,21,CANNES,18,28
480 DATA CHERBOURG,6,20,ROANNE,13,25,ARR
AS,4,24,TOURS,10,21,GRENOBLE,14,26,CHALO
NS-S-MARNE,7,25,NANCY,7,27
490 DATA LA ROCHELLE,13,20,CAEN,6,21,STR
ASBOURG,8,28,ST BRIEUC,7,18,BOURGES,11,2
3,QUIMPER,8,17,AMIENS,5,23
500 DATA LORIENT,9,18,ROUBAIX,4,24,BORDE
AUX,15,20,BELFORT,10,28,PERPIGNAN,19,24,
LE MANS,9,21,MARSEILLE,18,26
510 DATA LIMOGES,13,22,ROUEN,6,22,ORLEAN
S,9,23,LILLE,4,24,POITIER,12,21,MULHOUSE
,9,28,ST-ETIENNE,14,25
520 DATA PARIS,7,23,ANGOULEME,13,21,REIM
S,7,25,CLERMONT-FERRAND,13,24,METZ,7,27,
CHATEAUROUX,11,21,NEVERS,11,24
530 DATA RENNES,8,19,AGEN,16,21,BESANCON
,10,27,MONTAUBANT,17,22,CHAMBERY,13,26,B
LOIS,10,22,CHALON,11,26
540 DATA TOULOUSE,18,22,TROYES,8,25,ANGE
RS,10,20,ALBIT,17,23,DIJON,10,26,NIMES,1
7,25,MACON,12,26,VALENCE,15,26
550 DATA PERIGUEUX,15,21,LYON,13,26
560 REM *BOUCLE PRINCIPALE*
570 RESTORE 460
580 CALL HCHAR(3,4,32,12)
590 CALL HCHAR(21,4,32,20)
600 NX=INT(RND*65)+1
610 NES=NES+1
620 FOR M=1 TO NX
630 READ W$,Y,U
640 NEXT M
650 CALL HCHAR(21,3,32,23)
660 SC=0
670 P=4
680 A=21
690 CA(Y-2,U-17)=1
700 GOSUB 790
710 GOSUB 900
720 GOTO 570
730 REM * RESULTAT PARTIE *
740 CALL CLEAR
750 PRINT TAB(5);"LA PARTIE EST FINIE":
: : : : :TAB(5);"VOTRE POURCENTAGE":TAB(
5);"DE REUSSITE"
760 PRINT TAB(5);"EST DE :";(100*Q)/(NES
-1);"%": : : :"POUR REJOUER TAPER  RUN"
770 END
780 REM * DISPLAY AT *
790 FOR I=1 TO LEN(W$)
800 CALL HCHAR(A,P,ASC(SEG$(W$,I,1)))
810 P=P+1
820 IF P>23 THEN 850
830 NEXT I
840 RETURN
850 P=2
860 A=A+1
870 GOTO 830
880 REM * DEPLACEMENT ET *
890 REM * TEST CASE BONNE*
900 PO=0
910 S(1)=18
920 D(1)=17
930 S(2)=18
940 D(2)=17
950 F=32
960 CALL HCHAR(S(1),D(1),158)
970 CALL KEY(3,KEY,STATUS)
980 CALL HCHAR(S(1),D(1),159)
990 IF STATUS=0 THEN 960
1000 IF KEY=61 THEN 740
1010 IF KEY<>69 THEN 1030
1020 S(1)=S(1)-1
1030 IF KEY<>83 THEN 1050
1040 D(1)=D(1)-1
1050 IF KEY<>68 THEN 1070
1060 D(1)=D(1)+1
1070 IF KEY<>88 THEN 1090
1080 S(1)=S(1)+1
1090 CALL HCHAR(S(2),D(2),F)
1100 CALL GCHAR(S(1),D(1),F)
1110 CALL HCHAR(S(1),D(1),158)
1120 S(2)=S(1)
1130 D(2)=D(1)
1140 IF KEY<>13 THEN 960
1150 IF CA(S(1)-2,D(1)-17)=1 THEN 1310
1160 W$="ERREUR"
1170 CALL SOUND(200,-3,10)
1180 CALL HCHAR(3,3,32,10)
1190 P=4
1200 A=3
1210 GOSUB 790
1220 PO=PO+1
1230 SC=SC+1
1240 CALL HCHAR(A,P+1,158,PO)
1250 CALL HCHAR(S(1),D(1),F)
1260 IF SC<>SCD THEN 1300
1270 X=32
```

```
1280 GOSUB 1920
1290 RETURN
1300 GOTO 910
1310 W$="BRAVO"
1320 CALL HCHAR(3,3,32,10)
1330 Q=Q+1
1340 P=4
1350 A=3
1360 GOSUB 790
1370 FOR I=0 TO 300
1380 NEXT I
1390 CALL HCHAR(S(1),D(1),F)
1400 RETURN
1410 REM * TRACE CARTE *
1420 RESTORE 1500
1430 FOR W=33 TO 62
1440 IF W<>45 THEN 1460
1450 GOTO 1490
1460 READ Q$,E,R
1470 CALL CHAR(W,Q$)
1480 CALL HCHAR(E,R,W)
1490 NEXT W
1500 DATA 00001C2444548281,3,23,01020405
04040201,4,22,01070C10608,5,22,000000000
007080B,5,21
1510 DATA 13103E03FE00C0C,6,21,E09088A68
140404,6,20,404040402010F,7,20,0000000
00000C023,7,19
1520 DATA 0000001CE2010002,7,18,00000000
013EC08,7,17,8C6C2010E084700F,8,17,80402
41008040201,9,18
1530 DATA C04CCC807906784,10,19,20202020
10101008,11,19,0404020101010201,12,19,98
58202010100808,13,20
1540 DATA 08286454524A4986,14,20,0101030
202070503,15,19,0202020204040404,16,19,0
808080810101511,17,19
1550 DATA 32CC807808101E01,18,19,C020100
8040201,19,20,000000000003FC,19,21,00000
00000E01C03,19,22
1560 DATA C03807,20,23,1008F,20,24,08382
0100808C8C8,19,24,0000010204080808,18,24
1570 DATA D120DC03,18,25
1580 CALL CHAR(64,"03B35093330C0201")
1590 CALL HCHAR(18,26,64)
1600 FOR W=91 TO 155
1610 READ Q$,E,R
1620 CALL CHAR(W,Q$)
1630 CALL HCHAR(E,R,W)
1640 NEXT W
1650 DATA 00000000606304F8,18,27,0848102
0408,18,28,0101090206686808,17,28,204080
804020100E,16,28
1660 DATA 804040202010101,15,28,80601010
1020408,14,28,0403410204040281,13,27,202
0C898A4A4C484,12,27
1670 DATA 010106080810102,11,27,08088810
E080408,10,28,0808081008686808,9,28,E018
060102060808,7,28
1680 DATA 3434040408080808,8,28,0020D010
10080C03,6,27,80422518,6,26,808245493A02
0209,5,25
1690 DATA 8898680464620121,4,24,80402112
0C000C0C,5,23
1700 DATA 000000C3A3302808,6,22,080403,7
,22,106080710E0E2E08,7,23,00000000010608
1,6,23
1710 DATA 001E22C3,8,24,0000000090403008
,8,25,002C5A81,7,24,C0C0008044300804,7,2
5
1720 DATA C0C000000000C0C,7,27,1C0000000
0000606,8,19,011A1A04CC33,10,20,04040B13
20C0808,9,21
1730 DATA 0000000C0CC12618,10,21,0804070
80A080402,11,21,000000804020201,11,22,10
0E02020301,12,22
1740 DATA 0106083240807807,10,22,C460584
8292A241,11,23,0000000000303,13,21
1750 DATA 3030001028444281,9,23,24102824
24444242,11,24,8080804040402020,10,24,414
120201010101,12,24
1760 DATA 080810D0D008101,13,24,00008080
4040202,12,25,1050102020101 01,13,25,2020
58588080402,14,25
1770 DATA 02020A0201010101,12,21,8040487
00606,13,22
1780 DATA 0000000078860101,17,20,8080808
08080A08,12,26,06051C1A02020101,15,20,80
601116180A0807,16,21
1790 DATA 0030CE,16,22,8080B2484926201,1
7,22,0000037C801,17,23,02008877,15,21
1800 DATA 020201313304 04CA,17,25,0000000
0C1221C,17,26
1810 DATA 0101010202020202,16,25,90A0C08
0A080808,15,26,B48080838488 8B8B,14,26,00
000408D0D020C,10,27
1820 DATA 36496091120C,14,27,8182819995F
582B,13,26,090A0C1020A0204,11,26,0001020
204646404,10,26
1830 CALL GCHAR(7,22,G)
1840 CALL HCHAR(8,23,G)
1850 CALL HCHAR(18,22,128)
1860 CALL HCHAR(1,2,158,30)
1870 CALL HCHAR(24,2,158,30)
1880 CALL VCHAR(1,32,159,24)
1890 CALL VCHAR(1,2,159,24)
1900 RETURN
1910 REM * SOLUTION *
1920 FOR Z=18 TO Y STEP -1
1930 CALL HCHAR(Z+1,17,X)
1940 CALL GCHAR(Z,17,X)
1950 CALL HCHAR(Z,17,158)
1960 NEXT Z
1970 FOR Z=18 TO U
1980 CALL HCHAR(Y,Z-1,X)
1990 CALL GCHAR(Y,Z,X)
2000 CALL HCHAR(Y,Z,159)
2010 NEXT Z
```

```
2020 FOR TP=1 TO 50
2030 CALL HCHAR(Y,Z-1,158)
2040 CALL HCHAR(Y,Z-1,159)
2050 NEXT TP
2060 CALL HCHAR(Y,Z-1,X)
2070 RETURN
```

Version originale

```
100 REM *****************
110 REM COPYRIGHT 99 MAG-
120 REM            ET
130 REM LAURENT DESOBELLE
140 REM *****************
150 REM  ***************                  DEB
UT PARTIE                    ***************

160 CALL CLEAR
170 CALL CHAR(158,"1898FF3D3C3CE404")
180 CALL CHAR(159,"1819FFBC3C3C2720")
190 DIM CA(18,11)
200 CALL SCREEN(15)
210 FOR Q=1 TO 16
220 CALL COLOR(Q,16,15)
230 NEXT Q
240 Q=0
250 W$="CE PROGRAMME PERMET   DE TESTER
VOS          CONNAISSANCES EN       GEOGRA
PHIE"
260 P=2
270 A=15
280 GOSUB 930
290 F=0
300 DIM D(2)
310 DIM S(2)
320 PRINT "CHOISISSEZ VOTRE FORCE":"1=FO
RT":"2=MOYEN":"3=DEBUTANT"
330 CALL KEY(0,KEY,STATUS)
340 M=M+1
350 IF STATUS=0 THEN 330
360 CALL CLEAR
370 GOSUB 1540
380 SCD=KEY-48
390 RANDOMIZE M
400 R=INT(RND*10)+1
410 ON R GOTO 420,440,460,480,500,520,54
0,560,580,600
420 RESTORE 620
430 GOTO 730
440 RESTORE 630
450 GOTO 730
460 RESTORE 640
470 GOTO 730
480 RESTORE 650
490 GOTO 730
500 RESTORE 660
510 GOTO 730
520 RESTORE 670
530 GOTO 730
540 RESTORE 680
550 GOTO 730
560 RESTORE 690
570 GOTO 730
580 RESTORE 700
590 GOTO 730
600 RESTORE 710
610 REM ***************              FICH
IER VILLE              ***************
620 DATA CALAIS,3,23,MONTPELLIER,18,25,A
NNECY,13,27,BOULOGNE-S-M,4,22,TOULON,18,
27,BIARRITZ,17,19,NANTES,10,19
630 DATA BREST,8,17,VERSAILLES,7,23,CHAR
LEVILLE-MEZIERES,5,25,AIX-EN-PROVENCE,18
,26,LE HAVRE,5,21,CANNES,18,28
640 DATA CHERBOURG,6,20,ROANNE,13,25,ARR
AS,4,24,TOURS,10,21,GRENOBLE,14,26,CHALO
NS-S-MARNE,7,25,NANCY,7,27
650 DATA LA ROCHELLE,13,20,CAEN,6,21,STR
ASBOURG,8,28,ST BRIEUC,7,18,BOURGES,11,2
3,QUIMPER,8,17,AMIENS,5,23
660 DATA LORIENT,9,18,ROUBAIX,4,24,BORDE
AUX,15,20,BELFORT,10,28,PERPIGNAN,19,24,
LE MANS,9,21,MARSEILLE,18,26
670 DATA LIMOGES,13,22,ROUEN,6,22,ORLEAN
S,9,23,LILLE,4,24,POITIER,12,21,MULHOUSE
,9,28,ST-ETIENNE,14,25
680 DATA PARIS,7,23,ANGOULEME,13,21,REIM
S,7,25,CLERMONT-FERRAND,13,24,METZ,7,27,
CHATEAUROUX,11,21,NEVERS,11,24
690 DATA RENNES,8,19,AGEN,16,21,BESANCON
,10,27,MONTAUBANT,17,22,CHAMBERY,13,26,B
LOIS,10,22,CHALON,11,26
700 DATA TOULOUSE,18,22,TROYES,8,25,ANGE
RS,10,20,ALBIT,17,23,DIJON,10,26,NIMES,1
7,25,MACON,12,26,VALENCE,15,26
710 DATA PERIGUEUX,15,21,LYON,13,26,FIN,
1,1
720 REM *****************            BOUC
LE PRINCIPALE            ****************
**
730 FOR M=1 TO 66
740 READ W$,Y,U
750 IF W$<>"FIN" THEN 780
760 RESTORE 620
770 GOTO 850
780 CALL HCHAR(21,3,32,23)
790 SC=0
800 P=4
810 A=21
820 CA(Y-2,U-17)=1
830 GOSUB 930
840 GOSUB 1050
850 NEXT M
860 REM ****************             RESU
LTAT PARTIE            ****************

870 CALL CLEAR
880 PRINT TAB(5);"LA PARTIE EST FINIE":
```

```
: : : : :TAB(5);"VOTRE POURCENTAGE":TAB(
5);"DE REUSSITE"
890 Y=(100*Q)/(M-1)
900 PRINT TAB(5);"EST DE :";Y;"%": : : :
"POUR REJOUER TAPER  RUN"
910 END
920 REM *************                SP P
RINT AT                   *************
930 FOR I=1 TO LEN(W$)
940 E$=SEG$(W$,I,1)
950 O=ASC(E$)
960 CALL HCHAR(A,P,O)
970 P=P+1
980 IF P>23 THEN 1010
990 NEXT I
1000 RETURN
1010 P=2
1020 A=A+1
1030 GOTO 990
1040 REM *****************          SP
DEPLACEMENT ET             TEST CASE BONNE
            *****************
1050 PO=0
1060 S(1)=18
1070 D(1)=17
1080 S(2)=18
1090 D(2)=17
1100 F=32
1110 CALL HCHAR(S(1),D(1),158)
1120 CALL KEY(3,KEY,STATUS)
1130 CALL HCHAR(S(1),D(1),159)
1140 IF STATUS=0 THEN 1110
1150 IF KEY=69 THEN 1160 ELSE 1170
1160 S(1)=S(1)-1
1170 IF KEY=83 THEN 1180 ELSE 1190
1180 D(1)=D(1)-1
1190 IF KEY=68 THEN 1200 ELSE 1210
1200 D(1)=D(1)+1
1210 IF KEY=88 THEN 1220 ELSE 1230
1220 S(1)=S(1)+1
1230 CALL HCHAR(S(2),D(2),F)
1240 CALL GCHAR(S(1),D(1),F)
1250 CALL HCHAR(S(1),D(1),158)
1260 S(2)=S(1)
1270 D(2)=D(1)
1280 IF KEY=13 THEN 1290 ELSE 1110
1290 IF CA(S(1)-2,D(1)-17)=1 THEN 1450 E
LSE 1300
1300 W$="ERREUR"
1310 CALL SOUND(200,-3,10)
1320 CALL HCHAR(3,3,32,10)
1330 P=4
1340 A=3
1350 GOSUB 930
1360 PO=PO+1
1370 SC=SC+1
1380 CALL HCHAR(A,P+1,158,PO)
1390 CALL HCHAR(S(1),D(1),F)
1400 IF SC<>SCD THEN 1440
1410 X=32
1420 GOSUB 2040
1430 RETURN
1440 GOTO 1060
1450 W$="BRAVO"
1460 CALL HCHAR(3,3,32,10)
1470 Q=Q+1
1480 P=4
1490 A=3
1500 GOSUB 930
1510 CALL HCHAR(S(1),D(1),F)
1520 RETURN
1530 REM ***************              SP
TRACE CARTE                ET FICHIER
            ***************
1540 RESTORE 1620
1550 FOR W=33 TO 62
1560 IF W<>45 THEN 1580
1570 GOTO 1610
1580 READ Q$,E,R
1590 CALL CHAR(W,Q$)
1600 CALL HCHAR(E,R,W)
1610 NEXT W
1620 DATA 00001C2444548281,3,23,01020405
04040201,4,22,01070C1060800000,5,22,0000
00000007080B,5,21
1630 DATA 13103E03FE00C0C0,6,21,E09088A6
81404040,6,20,40404040402010F0,7,20,0000
00000000C023,7,19
1640 DATA 0000001CE2010002,7,18,00000000
013EC080,7,17,8C6C2010E084700F,8,17,8040
241008040201,9,18
1650 DATA C04CCC8079067840,10,19,2020202
010101008,11,19,0404020101010201,12,19,9
858202010100808,13,20
1660 DATA 08286454524A4986,14,20,0101030
202070503,15,19,0202020204040404,16,19,0
808080810101511,17,19
1670 DATA 32CC807808101E01,18,19,C020100
804020100,19,20,000000000003FC00,19,21,0
000000000E01C03,19,22
1680 DATA C038070000000000,20,23,1008F00
000000000,20,24,083820100808C8C8,19,24,0
000010204080808,18,24
1690 DATA D120DC0300000000,18,25
1700 CALL CHAR(64,"03B35093330C0201")
1710 CALL HCHAR(18,26,64)
1720 FOR W=91 TO 155
1730 READ Q$,E,R
1740 CALL CHAR(W,Q$)
1750 CALL HCHAR(E,R,W)
1760 NEXT W
1770 DATA 00000000606304F8,18,27,0848102
040800000,18,28,0101090206686808,17,28,2
04080804020100E,16,28
1780 DATA 8040402020101010,15,28,8060101
010204080,14,28,0403410204040281,13,27,2
020C898A4A4C484,12,27
1790 DATA 0101060808101020,11,27,0808881
```

```
0E0804080,10,28,0808081008686808,9,28,E0
18060102060808,7,28
1800 DATA 3434040408080808,8,28,0020D010
10080C03,6,27,8042251800000000,6,26,8082
45493A020209,5,25
1810 DATA 8898680464620121,4,24,80402112
0C000C0C,5,23
1820 DATA 000000C3A3302808,6,22,08040300
00000000,7,22,106080710E0E2E08,7,23,0000
000001060810,6,23
1830 DATA 001E22C300000000,8,24,00000000
90403008,8,25,002C5A8100000000,7,24,C0C0
008044300804,7,25
1840 DATA C0C000000000C0C0,7,27,1C000000
00000606,8,19,011A1A04CC330000,10,20,040
40B1320C08080,9,21
1850 DATA 0000000C0CC12618,10,21,0804070
80A080402,11,21,0000008040202010,11,22,1
00E020203010000,12,22
1860 DATA 0106083240807807,10,22,C460584
8292A2410,11,23,0000000000303000,13,21
1870 DATA 3030001028444281,9,23,24102824
24444242,11,24,8080804040402020,10,24,41
41202010101010,12,24
1880 DATA 080810D0D0081010,13,24,0000808
040402020,12,25,1050102020101010,13,25,2
020585880804020,14,25
1890 DATA 02020A0201010101,12,21,8040487
006060000,13,22
1900 DATA 0000000078860101,17,20,8080808
08080A080,12,26,06051C1A02020101,15,20,8
0601116180A0807,16,21
1910 DATA 0030CE0000000000,16,22,8080B24
849262010,17,22,0000037C80100000,17,23,0
200887700000000,15,21
1920 DATA 020201313330404CA,17,25,0000000
0C1221C00,17,26
1930 DATA 0101010202020202,16,25,90A0C08
0A0808080,15,26,B4808083848888B8B,14,26,0
0000408D0D020C0,10,27
1940 DATA 36496091120C0000,14,27,8182819
995F582B0,13,26,090A0C1020A02040,11,26,0
0010202046464O4,10,26
1950 CALL GCHAR(7,22,G)
1960 CALL HCHAR(8,23,G)
1970 CALL HCHAR(18,22,128)
1980 CALL HCHAR(1,2,158,30)
1990 CALL HCHAR(24,2,158,30)
2000 CALL VCHAR(1,32,159,24)
2010 CALL VCHAR(1,2,159,24)
2020 RETURN
2030 REM *************                  SP
SOLUTION                   *************
2040 FOR Z=18 TO Y STEP -1
2050 CALL HCHAR(Z+1,17,X)
2060 CALL GCHAR(Z,17,X)
2070 CALL HCHAR(Z,17,158)
2080 NEXT Z
2090 FOR Z=18 TO U
2100 CALL HCHAR(Y,Z-1,X)
2110 CALL GCHAR(Y,Z,X)
2120 CALL HCHAR(Y,Z,159)
2130 NEXT Z
2140 FOR TP=1 TO 50
2150 CALL HCHAR(Y,Z-1,158)
2160 CALL HCHAR(Y,Z-1,159)
2170 NEXT TP
2180 CALL HCHAR(Y,Z-1,X)
2190 RETURN
```

Une touche "RESET" sur le TI-99/4A

Jean-Luc Bazanegue

ATTENTION : Toute intervention sur le TI-99/4A entraîne la cessation de la garantie accordée par Texas Instruments France. Ceux qui effectuent la modification indiquée ci-dessous le font donc à leurs risques et périls.

Tous ceux qui utilisent un TI-99 se sont aperçu, ou s'apercevront un jour, que la touche "RESET", présente sur de nombreux micro-ordinateurs, fait parfois cruellement défaut. "99 Magazine" vous propose donc de combler par vous-même cette lacune. Avant "d'opérer", voyons à quoi peut nous servir cette touche.



Dans certains cas, pour reprendre le contrôle de l'ordinateur, il faut le mettre hors tension. Cette solution n'est pas très élégante et peut provoquer des réactions imprévisibles du côté des périphériques. De plus, lors de la mise au point d'un programme en assembleur, le fait de couper l'alimentation détruit tout ce qui se trouvait en mémoire au moment où l'ordinateur a repris son indépendance. La touche "RESET" laisse de grandes chances de détecter une

"bogue" éventuelle, en allant fouiller dans la mémoire.

Une pression sur cette touche donne apparemment le même résultat que "QUIT" en Basic, mais des différences importantes séparent ces deux systèmes. La touche "QUIT" (FCNT =) fonctionne par logiciel. A chaque fois que l'interpréteur exécute une ligne de programme, il scrute le clavier afin de déterminer si cette touche est enfoncée. Le temps de réponse est souvent relativement long et cette méthode n'est pas toujours efficace. La touche que nous vous proposons d'installer sur votre TI agit directement sur le microprocesseur et son temps de réponse est de trois cycles d'horloge, soit environ une microseconde !

Ce bricolage très simple ne requiert pas de connaissances en électronique. Il faut souder deux fils sur la carte principale du TI aux endroits indiqués sur le schéma. Ils devront être raccordés côté soudures. Ceci n'est pas très rationnel, mais l'implantation des éléments existants empêche la connection côté composants. Le circuit étant recouvert d'un verni isolant, les points destinés à recevoir les soudures devront être soigneusement décapés. Les autres extrémités de ces fils seront connectés à un bouton poussoir unipolaire (notre touche RESET) que vous pourrez fixer à l'arrière de la console.

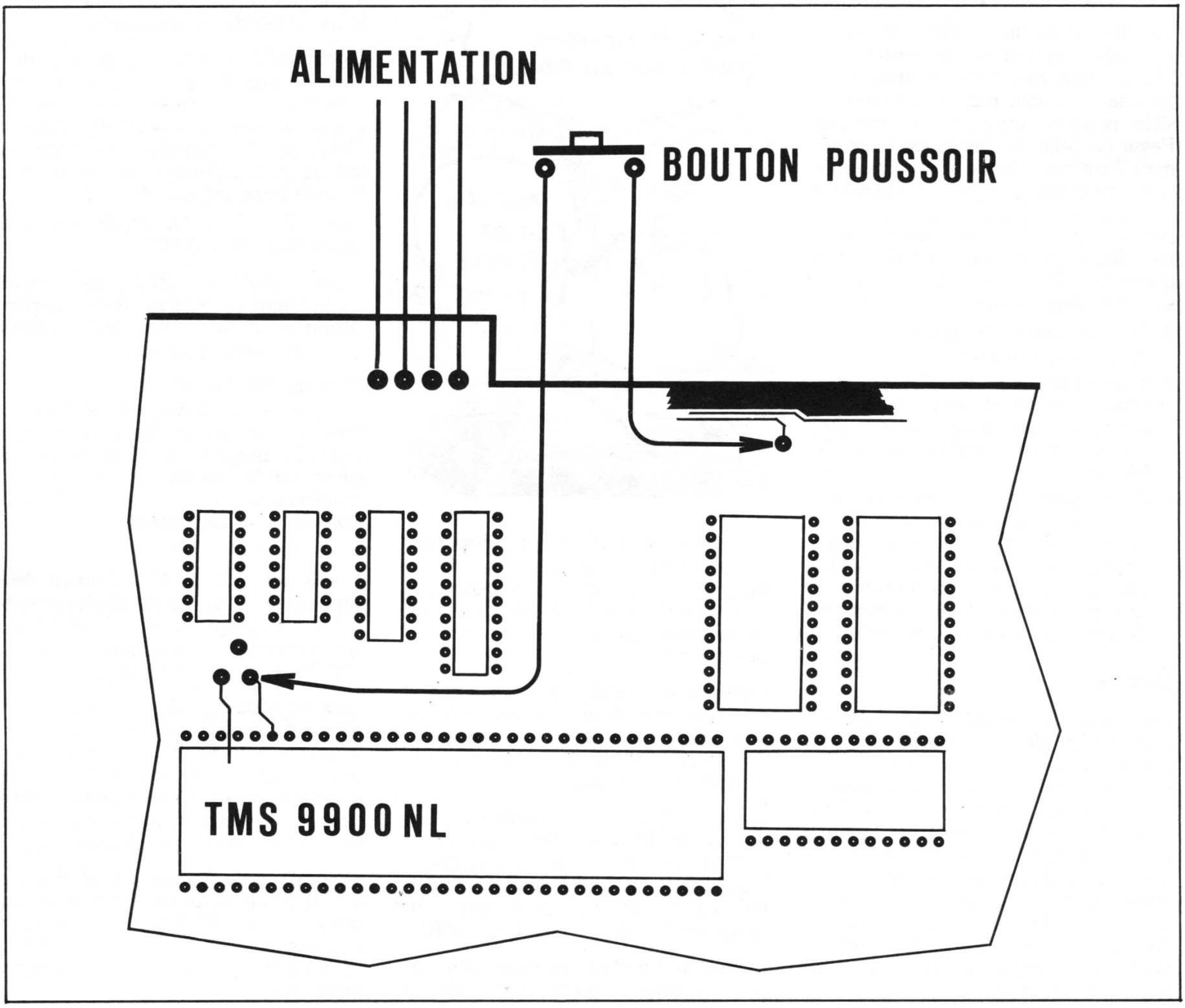
Le démontage du TI ne pose pas de problème particulier. Les étapes à respecter sont les suivantes :

- retirer le bouton Marche/Arrêt en le tirant vers l'avant ;
- enlever les vis qui maintiennent le fond de la console ;
- enlever l'alimentation (circuit placé à côté du clavier) ;
- séparer les deux connecteurs qui relient l'alimentation à la carte principale ;
- ôter les deux vis qui maintiennent la carte principale ;
- enlever cette carte en débranchant le câble plat qui la relie au clavier ;
- extraire le connecteur des cartouches et retirer les tôles de blindage.

Vous êtes maintenant prêt à opérer. Les soudures ne font normalement pas courir de risques à votre cher TI, sous réserve de prendre quelques précautions indispensables :

- munissez-vous d'un fer à souder à panne très fine ;
- le circuit doit chauffer le moins possible. Les soudures doivent donc être exécutées rapidement ;
- certains fers à souder laissent s'échapper des courants de fuite importants, il est donc plus prudent de débrancher le fer juste avant d'effectuer les soudures ;
- enfin, si vous avez quelques doutes sur vos capacités manuelles, il vaut mieux renoncer ou confier cette tâche à un ami plus bricoleur.

Le schéma représente le circuit **COTE SOUDURES.**



Puzzle

Marianne Sutz

Ce programme, écrit en TI Basic, ne prétend pas égaler en définition et en nombre de pièces ses honorables prédécesseurs en papier ou en carton. Par contre, le risque de ne pouvoir terminer le puzzle parce qu'une pièce s'est glissée sous la bibliothèque est beaucoup moins grand ! Le puzzle que je vous propose est constitué de 48 pièces elles-mêmes composées de deux caractères graphiques superposés. Pour qu'il puisse être utilisé immédiatement, ce programme comporte un dessin stocké sous forme de DATAs mais vous pourrez créer et sauvegarder vos propres dessins sur disquette et/ou sur cassette. Nous verrons cela un peu plus loin.

Principes du jeu

Les 48 pièces du puzzle sont réparties aléatoirement sur la gauche de l'écran. Une fois cette répartition effectuée, le dessin reconstitué sera visible pendant environ une minute. Passé ce délai, le dessin est masqué, mais il est possible de le faire réapparaître en cours de partie en appuyant sur la touche "égal" (=). Les déplacements de pièces se font en donnant leurs coordonnées de départ et d'arrivée. Quelques précisions au sujet des déplacements :

- ils s'effectuent toujours vers le puzzle à reconstituer ;
- si une pièce est envoyée vers une position occupée, elle prendra la place de l'autre pièce qui sera elle-même reportée sur la gauche de l'écran ;
- il est possible de ramener une pièce du puzzle vers la gauche de l'écran en envoyant une pièce "inexistante" (coordonnées de départ et d'arrivée correspondant à une position vide) sur la position de celle que vous voulez déplacer.

Score

En début de partie, vous disposez d'un capital de 100 points. Chaque déplacement de pièce vous coûte un point et le fait de consulter le dessin original coûte trois points. Il faut donc essayer de reconstruire le puzzle en effectuant un minimum de déplacements. Lorsque vous aurez terminé, votre score sera affiché d'une manière symbolique, car la redéfinition des caractères ne permet pas un affichage du score "en clair" (le 8 et le 9 sont redéfinis).

Analyse du programme

Ce programme occupe une place importante en mémoire et il ne m'a pas été possible d'insérer des REMarques (sauf pour le titre). Une explication des points principaux est donc nécessaire.

Les tableaux

Le tableau T1 sert uniquement à la répartition aléatoire des pièces. Le tableau T2 permet au programme de savoir si le puzzle est reconstitué. Les tableaux TD et TA sont utilisés pour les mouvements de pièces. Le tableau TA est aussi utilisé en liaison avec le tableau T2 pour déterminer si le puzzle est reconstitué.



Le tableau TC$ n'est pas indispensable dans la version de base de ce programme ; il n'est là que pour faciliter l'implantation du segment de programme qui permet le stockage des dessins.

Lignes 640 à 660 : tous les groupes de caractères sont mis en vert sur vert. Ceci permet de "masquer" l'écran pendant les initialisations et la répartition des pièces.

Lignes 670 à 830 : redéfinition des caractères 32 à 47, 56 à 64, 86 à 159 et 75. Le caractère 32 (espace) étant redéfini, l'instruction CALL HCHAR(1,1,86,768) de la ligne 830 remplace l'habituel "CALL CLEAR".

Lignes 840 à 940 : la ligne 840 appelle le sous-programme qui affiche le puzzle reconstitué. Les lignes suivantes assurent de la répartition aléatoire des pièces et initialisent le tableau TD.

Lignes 950 à 1400 : affichage des coordonnées et du texte. Le sous-programme situé à la ligne 2890 permet la simulation de "DISPLAY AT" (Basic Etendu). Les lignes 1130 à 1220 donnent leurs couleurs définitives à l'écran et aux groupes de caractères.

Lignes 1410 à 1430 : délai avant le masquage du dessin. La partie débute après l'exécution de cette boucle (environ une minute).

Lignes 1440 à 1960 : saisie des coordonnées de départ et d'arrivée et mise à jour des tableaux. Les lignes 1490 et 1750 permettent une utilisation du programme quelle que soit la position de la touche "ALPHA LOCK".

Lignes 1970 à 2250 : déplacement des pièces et comparaison du puzzle avec le tableau T2. Si les tableaux T2 et TA contiennent les mêmes valeurs, le puzzle est reconstitué.

Lignes 2260 à 2600 : fin de la partie. Celle-ci s'arrête si le puzzle est reconstitué ou si vous n'avez plus de point. Dans les lignes 2480, 2510 et 2540, le "V" remplace l'espace du fait de la redéfinition des caractères. Il aurait aussi été possible d'écrire :

2480 T$="PLUS"&CHR$(86)&"DE"&CHR$(86)&"POINT".

Lignes 2610 à 3270 : ces lignes contiennent la totalité des sous-programmes appelés par les routines que nous venons de voir.

Version "fichiers"

Elle permet la sauvegarde et l'utilisation de dessins sur disquette ou cassette. Le programme n'est pas enregistré sur la cassette N° 3, mais les modifications à apporter sur la version de base sont très simples à réaliser.

Les lignes 260 à 470 doivent être supprimées et remplacées par le segment de programme "FICHIERS". Ce segment ne peut pas "tourner" sans le programme de base.

Lors du stockage de vos dessins, les pièces du puzzle sont numérotées de la gauche vers la droite et de haut en bas. Il faut prendre garde de ne pas avoir de pièces identiques car, dans ce cas, la reconstitution du puzzle pourrait poser des problèmes.

La gestion de fichiers en TI Basic a été abordée dans les deux premiers numéros de "99 Magazine", je vous propose donc de vous y reporter si vous désirez quelques détails supplémentaires.

```
100 REM  ******************
110 REM  *      PUZZLE      *
120 REM  ******************
130 REM  *    BASIC TI      *
140 REM  ******************
150 REM  *   COPYRIGHT     *
160 REM  *  99 MAGAZINE    *
170 REM  *       ET         *
180 REM  *    MARIANNE     *
190 REM  *      SUTZ        *
200 REM  ******************
210 CALL CLEAR
220 RANDOMIZE
230 OPTION BASE 1
240 DIM T1(48),T2(48),TD(8,6),TA(6,8),TC
$(96)
250 P=100
260 DATA FF8681809CC2818,8080809E8181808
,FF0080601008OB84,700E010000128904,FF000
000000806
270 DATA 1008040404080886,FF00000000003
04,08122414222O103F,FF00010E1023C4,010A1
410204020F
280 DATA FF00DF2006800F7,80000000000000
1,FF007887000031CE,00003845020000 8,FF3FC1
011FE10101
290 DATA 0101E11101010101,80808080908884
84,8484888888888909,00100808081,000004048
4424242
300 DATA 4140080585422122,21282444090902
04,20A0A03F1010101,1090909O1010101,10101
0F02020202
310 DATA 212224283020408,030204081020408
,00000000007F417F,C040201008040201,00000
00000FE82FE
320 DATA 0101010101010101,81412111090503
01,8080808180808484,8883848892818182,424
2840481
330 DATA 00E39C402010101,020202010102040
8,F1020408102040FF,11121418102040 8,00000
000000000FF
340 DATA 0,00000000000000FF,417F417F7F7F
7F7F,7F7F7F7F7F0000FF,82FE82FEFEFEFEFE
350 DATA FEFEFEFEFE0000FF,01010101010101
01,01010101010101FF,8484B880B8C78484,848
48484828292AA
360 DATA 214284408,0,407F404040404041,5F
515F515F515F5F,00FF0000000000FF
370 DATA FF04040404FF0404,00FF000000000
F,FF111F111FF11F1F,00FF0000007F404,7F202
0202020202
380 DATA 00FF000000FE0202,FE040404040404
04,01FF01010101010F,FF89F989F98FF9F9,AAA
B938383858585
390 DATA 878585898E88C4A4,002424FF242424
24,FF277C8000000304,5FDFDFDFDFDFDFDF,C1C
04040404C52E1
400 DATA 0404FF04040404FF,FF0000000000000
1A,1F1FFF1F1F1F1FFF,F000000000001A,202
020202020202
410 DATA 2020202020202202,04040404141404O
4,0404040404040404,F9F9FFF9F9F9F9FF,0F01
0101010101A1
420 DATA A4A4A49485848484,8282928A899080
FF,0040C1B20A04,00000000008040FF,0000808
0A0B44A21
430 DATA 00000000000001FF,E5000000004060
5,CC020100031CE0FF,AA54000186601,00030C7
0800000FF
440 DATA FF40407F4040FF4,8000000000000F
F,FF0202FE0202FF02,0204040408080 8FF,5F01
051921418101,01010101010101FF"
450 FOR B=1 TO 96
460 READ TC$(B)
470 NEXT B
480 PRINT TAB(10);"**********":TAB(10);"
*          *":TAB(10);"* PUZZLE *":TAB(10)
;"*          *"
490 PRINT TAB(10);"**********"::"le but
de ce jeu est la      reconstitution d'un
 puzzle a"
500 PRINT "l'aide des pieces reparties a
leatoirement sur la gauche"
510 PRINT "de l'ecran.":"lorsque ces pie
ces seront   affichees, vous diposerez"
520 PRINT "d'environ une minute pour    m
emoriser le dessin."
530 PRINT "le dessin sera alors masque e
t vous devrez reconstituer le puzzle."::
::
540 GOSUB 3190
550 PRINT "vous disposez d'un capital   d
e 100 points."
560 PRINT "chaque deplacement de piece v
ous enleve un point."
570 PRINT "il est possible de demasquerl
e dessin pour environ 30"
580 PRINT "secondes en appuyant sur la t
ouche = mais ceci vous"
590 PRINT "fait perdre 3 points.        s
i vous reussissez a"
600 PRINT "reconstituer le puzzle avantl
'epuisement de vos points,"
610 PRINT "votre score sera affiche      s
ous une forme symbolique."::
620 PRINT "apres la lecture de cette    p
age, vous devrez attendre  une minute.":
::
630 GOSUB 3190
640 FOR B=1 TO 16
650 CALL COLOR(B,4,4)
660 NEXT B
670 DB=32
680 FB=47
690 I=1
700 GOSUB 2610
710 DB=56
720 FB=63
```

```
730 I=9
740 GOSUB 2610
750 DB=88
760 FB=159
770 I=13
780 GOSUB 2610
790 CALL CHAR(86,"0")
800 CALL CHAR(64,"FFFFFFFFFFFFFFFF")
810 CALL CHAR(75,"00182442422418")
820 CALL CHAR(87,"080C0EFF0E0C08")
830 CALL HCHAR(1,1,86,768)
840 GOSUB 2690
850 FOR B1=2 TO 23 STEP 3
860 FOR B2=2 TO 12 STEP 2
870 N=INT(48*RND)+1
880 IF T2(N)=0 THEN 870
890 CALL HCHAR(B1,B2,T2(N))
900 CALL HCHAR(B1+1,B2,T2(N)+1)
910 TD((B1+1)/3,B2/2)=T2(N)
920 T2(N)=0
930 NEXT B2
940 NEXT B1
950 C=48
960 FOR B=2 TO 12 STEP 2
970 CALL HCHAR(1,B,C)
980 C=C+1
990 NEXT B
1000 FOR B=16 TO 23
1010 CALL HCHAR(12,B,B+32)
1020 NEXT B
1030 C=65
1040 FOR B=2 TO 23 STEP 3
1050 CALL HCHAR(B,13,C)
1060 C=C+1
1070 NEXT B
1080 C=65
1090 FOR B=13 TO 23 STEP 2
1100 CALL HCHAR(B,15,C)
1110 C=C+1
1120 NEXT B
1130 CALL COLOR(1,13,16)
1140 CALL COLOR(2,13,16)
1150 CALL COLOR(3,2,8)
1160 CALL COLOR(4,13,16)
1170 FOR B=5 TO 7
1180 CALL COLOR(B,2,8)
1190 NEXT B
1200 FOR B=8 TO 16
1210 CALL COLOR(B,13,16)
1220 NEXT B
1230 CALL SCREEN(8)
1240 L=1
1250 C=16
1260 T$="DEPART"
1270 GOSUB 2890
1280 L=2
1290 T$="LIGNE"
1300 GOSUB 2890
1310 L=6
1320 GOSUB 2890
1330 L=3
1340 T$="COLONNE"
1350 GOSUB 2890
1360 L=7
1370 GOSUB 2890
1380 L=5
1390 T$="DESTINATION"
1400 GOSUB 2890
1410 FOR D=0 TO 11111
1420 NEXT D
1430 GOSUB 2950
1440 CALL HCHAR(2,15,87)
1450 GOSUB 3240
1460 IF K<>61 THEN 1490
1470 GOSUB 3010
1480 GOTO 1440
1490 IF (K>64)*(K<73)+(K>96)*(K<105)THEN
 1520
1500 GOSUB 3150
1510 GOTO 1450
1520 IF K<96 THEN 1540
1530 K=K-32
1540 CALL HCHAR(2,23,K)
1550 LD=((K-64)*3)-1
1560 I1D=K-64
1570 CALL HCHAR(2,15,86)
1580 CALL HCHAR(3,15,87)
1590 GOSUB 3240
1600 IF K<>61 THEN 1630
1610 GOSUB 3010
1620 GOTO 1440
1630 IF (K>47)*(K<54)THEN 1660
1640 GOSUB 3150
1650 GOTO 1590
1660 CALL HCHAR(3,25,K)
1670 CD=(K-47)*2
1680 I2D=K-47
1690 CALL HCHAR(3,15,86)
1700 CALL HCHAR(6,15,87)
1710 GOSUB 3240
1720 IF K<>61 THEN 1750
1730 GOSUB 3010
1740 GOTO 1440
1750 IF (K>64)*(K<71)+(K>96)*(K<103)THEN
 1780
1760 GOSUB 3150
1770 GOTO 1710
1780 IF K<96 THEN 1800
1790 K=K-32
1800 CALL HCHAR(6,23,K)
1810 LA=11+((K-64)*2)
1820 I1A=K-64
1830 CALL HCHAR(6,15,86)
1840 CALL HCHAR(7,15,87)
1850 GOSUB 3240
1860 IF S=0 THEN 1850
1870 IF K<>61 THEN 1900
1880 GOSUB 3010
```

```
1890 GOTO 1440
1900 IF (K>47)*(K<56)THEN 1930
1910 GOSUB 3150
1920 GOTO 1850
1930 CALL HCHAR(7,25,K)
1940 CA=K-32
1950 I2A=K-47
1960 CALL HCHAR(7,15,86)
1970 IF TD(I1D,I2D)<>86 THEN 2000
1980 DR=0
1990 GOTO 2010
2000 DR=1
2010 IF (TA(I1A,I2A)=0)+(TA(I1A,I2A)=86)
THEN 2040
2020 IN=TA(I1A,I2A)
2030 GOTO 2050
2040 IN=86
2050 IF DR THEN 2080
2060 CALL VCHAR(LA,CA,86,2)
2070 GOTO 2100
2080 CALL HCHAR(LA,CA,TD(I1D,I2D))
2090 CALL HCHAR(LA+1,CA,TD(I1D,I2D)+1)
2100 TA(I1A,I2A)=TD(I1D,I2D)
2110 IF IN=86 THEN 2160
2120 CALL HCHAR(LD,CD,IN)
2130 CALL HCHAR(LD+1,CD,IN+1)
2140 TD(I1D,I2D)=IN
2150 GOTO 2200
2160 CALL VCHAR(LD,CD,86,2)
2170 TD(I1D,I2D)=0
2180 P=P-1
2190 IF P<=0 THEN 2290
2200 I=1
2210 FOR V1=1 TO 6
2220 FOR V2=1 TO 8
2230 IF TA(V1,V2)=T1(I)THEN 2260
2240 GOSUB 3100
2250 GOTO 1440
2260 I=I+1
2270 NEXT V2
2280 NEXT V1
2290 FOR B=1 TO 7
2300 CALL HCHAR(B,15,86,13)
2310 NEXT B
2320 IF P<=0 THEN 2460
2330 GOSUB 2690
2332 IF P>1 THEN 2340
2334 T$="POINT"
2336 GOTO 2350
2340 T$="POINTS"
2350 L=1
2360 C=16
2370 GOSUB 2890
2380 FOR LI=3 TO 6
2390 FOR CO=16 TO 31
2400 CALL HCHAR(LI,CO,75)
2410 P=P-1
2420 IF P=0 THEN 2500
2430 NEXT CO
2440 NEXT LI
2450 GOTO 2500
2460 L=1
2470 C=16
2480 T$="PLUSVDEVPOINT"
2490 GOSUB 2890
2500 L=7
2510 T$="ARRETVDU"
2520 GOSUB 2890
2530 L=8
2540 T$="PROGRAMMEVPAR"
2550 GOSUB 2890
2560 L=9
2570 T$="ENTER"
2580 GOSUB 2890
2590 GOSUB 3200
2600 END
2610 FOR B=DB TO FB
2620 IF INT(B/2)<>B/2 THEN 2650
2630 T1(I)=B
2640 I=I+1
2650 I2=I2+1
2660 CALL CHAR(B,TC$(I2))
2670 NEXT B
2680 RETURN
2690 I=1
2700 FOR B1=13 TO 23 STEP 2
2710 FOR B2=25 TO 32
2720 CALL HCHAR(B1,B2,T1(I))
2730 T2(I)=T1(I)
2740 CALL HCHAR(B1+1,B2,T1(I)+1)
2750 I=I+1
2760 NEXT B2
2770 NEXT B1
2780 C=65
2790 FOR B=13 TO 23 STEP 2
2800 CALL HCHAR(B,24,C)
2810 C=C+1
2820 NEXT B
2830 C=48
2840 FOR B=25 TO 32
2850 CALL HCHAR(12,B,C)
2860 C=C+1
2870 NEXT B
2880 RETURN
2890 PC=1
2900 FOR B=C TO C+LEN(T$)-1
2910 CALL HCHAR(L,B,ASC(SEG$(T$,PC,1)))
2920 PC=PC+1
2930 NEXT B
2940 RETURN
2950 FOR B=25 TO 32
2960 CALL VCHAR(13,B,64,12)
2970 NEXT B
2980 CALL HCHAR(12,25,86,8)
2990 CALL VCHAR(13,24,86,12)
3000 RETURN
3010 CALL VCHAR(2,15,86,6)
3020 P=P-3
```

```
3030 IF P<=0 THEN 2290
3040 GOSUB 3100
3050 GOSUB 2690
3060 FOR D=0 TO 5000
3070 NEXT D
3080 GOSUB 2950
3090 RETURN
3100 CALL HCHAR(2,23,86)
3110 CALL HCHAR(3,25,86)
3120 CALL HCHAR(6,23,86)
3130 CALL HCHAR(7,25,86)
3140 RETURN
3150 CALL SOUND(10,440,0)
3160 CALL SOUND(10,294,0)
3170 CALL SOUND(10,392,0)
3180 RETURN
3190 PRINT TAB(5);"APPUYEZ SUR <ENTER>":
TAB(7);"POUR CONTINUER"
3200 CALL KEY(0,K,S)
3210 IF K<>13 THEN 3200
3220 CALL CLEAR
3230 RETURN
3240 CALL SOUND(10,500,0)
3250 CALL KEY(0,K,S)
3260 IF S=0 THEN 3250
3270 RETURN
```

Segment Fichiers

```
260 REM *****************
261 REM *SEGMENT FICHIERS*
262 REM *****************
263 SI$(1)="superieure"
264 SI$(2)="inferieure"
265 PRINT "PUZZLE": :"1-creation d'un fi
chier": :"2-utilisation d'un fichier": :
"3-fin du programme"
266 PRINT : :"ENTREZ VOTRE CHOIX..."
267 CALL KEY(0,K,S)
268 IF (K<49)+(K>51)THEN 267
269 IF K=51 THEN 400
270 IF K=50 THEN 342
272 FOR PIE1=1 TO 95 STEP 2
274 FOR PIE2=0 TO 1
276 CALL CLEAR
278 INPUT "partie "&SI$(PIE2+1)&" piece
"&STR$(PIE1-(1*(PIE1-1)/2))&"    ":TC$(PI
E1+PIE2)
280 LN=LEN(TC$(PIE1+PIE2))
282 IF LN<=16 THEN 292
284 PRINT :"CHAINE TROP LONGUE"
285 GOSUB 3150
286 FOR D=0 TO 500
288 NEXT D
290 GOTO 276
292 FOR B=1 TO LN
294 VL=ASC(SEG$(TC$(PIE1+PIE2),B,1))
296 IF (VL>47)*(VL<58)+(VL>64)*(VL<71)TH
EN 306
297 GOSUB 3150
298 PRINT :"le caractere "&CHR$(VL):"n'e
st pas valide"
304 GOTO 286
306 NEXT B
308 NEXT PIE2
310 NEXT PIE1
312 CALL CLEAR
314 PRINT "SAUVEGARDE DU FICHIER": :"1-s
ur cassette": :"2-sur disquette": :"ENTR
EZ VOTRE CHOIX..."
316 CALL KEY(0,K,S)
318 IF (K<49)+(K>50)THEN 316
319 CALL CLEAR
320 IF K=49 THEN 328
322 INPUT "nom du puzzle? ":NP$
324 OPEN #1:"DSK1."&NP$,OUTPUT,INTERNAL,
FIXED 17
326 GOTO 330
328 OPEN #1:"CS1",INTERNAL,OUTPUT,FIXED
330 FOR B=1 TO 96
332 PRINT #1:TC$(B)
334 NEXT B
336 CLOSE #1
338 CALL CLEAR
340 GOTO 264
342 CALL CLEAR
344 PRINT "1-fichier sur cassette": :"2-
fichier sur disquette": :"ENTREZ VOTRE C
HOIX..."
346 CALL KEY(0,K,S)
348 IF (K<49)+(K>50)THEN 346
349 CALL CLEAR
350 IF K=49 THEN 358
352 INPUT "nom du puzzle? ":NP$
354 OPEN #1:"DSK1."&NP$,INPUT ,INTERNAL,
FIXED 17
356 GOTO 360
358 OPEN #1:"CS1",INPUT ,INTERNAL,FIXED
360 FOR B=1 TO 96
362 INPUT #1:TC$(B)
364 NEXT B
366 CALL CLEAR
368 GOTO 480
400 CALL CLEAR
402 END
```

Réparation des boutons de joysticks

Ce "truc" nous est communiqué par Monsieur Jacques Vermont, un de nos fidèles lecteurs, de Saint-Nom-La-Bretèche.

Les pannes de joysticks proviennent souvent du fait que le contact du bouton poussoir est uniquement assuré par une couche de peinture conductrice. Pour remettre ce poussoir en état, il suffit de vaporiser un peu de peinture graphitée KF "blindotub" sur la face qui assure le contact du bouton.

Awari

Julien Thomas

Ce programme de jeu, écrit en **Basic TI**, nous vient de nos amis belges du club TISOFT-HOME. Nous lui avons apporté quelques modifications pour le rendre plus fiable et nous l'avons traduit en français.

Awari est un très ancien jeu africain dont l'origine exacte et l'époque d'apparition sont mal connues. Son nom et ses règles peuvent varier sensiblement d'un pays à l'autre.

Présentation

Le jeu se présente sous la forme d'une planche dans laquelle sont creusés 18 trous (ou cases) répartis de la façon suivante :

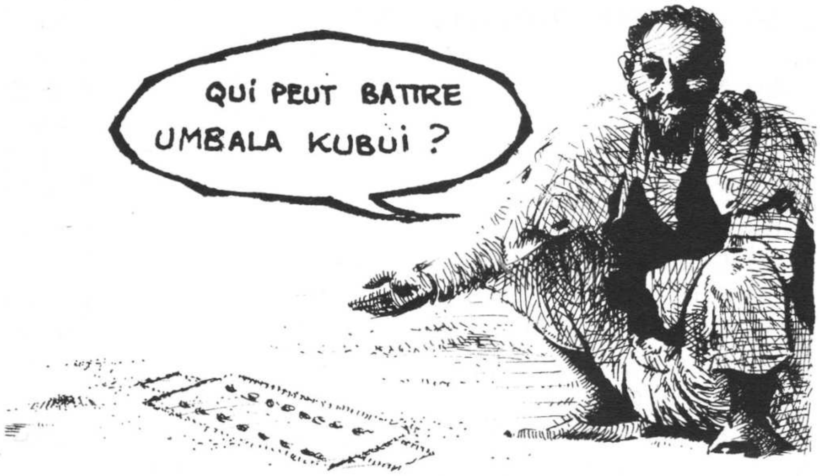
- 8 trous de chaque côté de la planche ;
- 2 trous à chaque extrémité : ce sont les cases personnelles des joueurs.

Les règles

En début de partie, les cases placées de chaque côté de la planche contiennent chacune 3 graines. La partie se termine lorsque toutes les cases situées sur un côté de la planche sont vides. Le vainqueur est le joueur qui a engrangé le plus de graines dans sa case personnelle.

Toute la stratégie du jeu réside dans le déplacement des graines, qui se fait de la manière suivantes :

- le choix de la case dans laquelle se trouvent les graines que vous voulez déplacer s'effectue à l'aide des touches "flèche à gauche" et "flèche à droite" (FCTN-S et FCTN-D). Ce choix doit être confirmé par une pression sur la touche ENTER ;
- toutes les graines situées dans la case choisie sont distribuées une par une dans les cases suivantes, et ce dans le sens inverse des aiguilles d'une montre ;
- si, lors du déplacement, une ou plusieurs graines arrivent au bout de la planche, le déplacement continue en passant par la case personnelle située à cette extrémité ;
- lorsque la dernière graine d'un déplacement arrive dans la case personnelle du joueur, celui-ci doit rejouer ;
- si la dernière graine du déplacement arrive dans une case vide directement opposée à une case occupée, toutes les graines situées dans la case occupée plus la dernière graine du déplacement sont reportées dans la case personnelle du joueur concerné.



Votre adversaire est le légendaire "UMBALA KUBUI" (alias TI-99/4A) et vous pourrez vour apercevoir que sa réputation n'est pas surfaite car il n'est possible de le battre qu'après avoir pratiqué ce jeu pendant un certain temps.

```
10 REM ***************
20 REM *    AWARI    *
30 REM ***************
40 REM *  BASIC  TI  *
50 REM ***************
60 REM *  COPYRIGHT  *
70 REM *  TISOFT ET  *
80 REM * 99 MAGAZINE *
90 REM ***************
150 CALL CLEAR
200 CALL COLOR(16,11,11)
250 CALL CHAR(152,"")
300 CALL CHAR(113,"6D6FFF7FFFFF793F")
350 CALL CHAR(114,"00010103071E3C7C")
400 CALL CHAR(115,"7FC77361811F3F7F")
450 CALL CHAR(116,"FF7F7C3E3F3F3F3F")
500 CALL CHAR(117,"EFC707070783C3E3")
550 CALL CHAR(118,"1F0F0F0703030101")
600 CALL CHAR(119,"E3F37B7F3F9F9FCF")
650 CALL CHAR(120,"0100000101037F7E")
700 CALL CHAR(121,"CFE7E773F0F8F838")
750 CALL CHAR(122,"B6F6FFFEFFFF9EFC")
800 CALL CHAR(123,"008080C0E0783C3E")
850 CALL CHAR(124,"FEE3CE8681F8FCFE")
900 CALL CHAR(125,"FFFE3E7CFCFCFCFC")
950 CALL CHAR(126,"F7E3E0E0E0C1C3C7")
1000 CALL CHAR(127,"F8F0F0E0C0C08080")
1050 CALL CHAR(128,"C7CFDEFEFCF9F9F3")
1100 CALL CHAR(129,"80000080800C0FE7E")
1150 CALL CHAR(131,"")
1200 CALL CHAR(130,"F3E7E7CE0F1F1F1C")
1250 GOSUB 1400
1300 GOSUB 2500
1350 GOTO 3050
1400 CALL CLEAR
1450 CALL HCHAR(16,15,131)
1500 CALL HCHAR(16,18,131)
1550 CALL HCHAR(16,16,113)
1600 CALL HCHAR(17,15,114)
1650 CALL HCHAR(17,16,115)
1700 CALL HCHAR(18,15,116)
1750 CALL HCHAR(18,16,117)
1800 CALL HCHAR(19,15,118)
1850 CALL HCHAR(19,16,119)
1900 CALL HCHAR(20,15,120)
1950 CALL HCHAR(20,16,121)
2000 CALL HCHAR(16,17,122)
2050 CALL HCHAR(17,18,123)
```

```
2100 CALL HCHAR(17,17,124)
2150 CALL HCHAR(18,18,125)
2200 CALL HCHAR(18,17,126)
2250 CALL HCHAR(19,18,127)
2300 CALL HCHAR(19,17,128)
2350 CALL HCHAR(20,18,129)
2400 CALL HCHAR(20,17,130)
2450 RETURN
2500 FOR Y=1 TO 3
2550 CALL SOUND(100,110,0,111,1,-5,8)
2600 CALL SOUND(400,110,30)
2650 FOR X=1 TO 3
2700 CALL SOUND(100,110,2,-5,15)
2750 CALL SOUND(400,110,30)
2800 NEXT X
2850 NEXT Y
2900 RETURN
3050 GOSUB 2500
3100 FOR X=1 TO 100
3150 NEXT X
3160 CALL CLEAR
3200 FOR X=9 TO 13
3250 CALL COLOR(X,2,16)
3300 NEXT X
3350 CALL CLEAR
3360 PRINT "AWARI EST UN ANCIEN JEU
AFRICAIN. IL SE PRATIQUE    AVEC 36 GRAI
NES, 6 CASES DE"
3370 PRINT "CHAQUE COTE ET DEUX CASES PE
RSONNELLES."
3380 PRINT :"VOUS DEVEZ DEPLACER TOUTES
 LES GRAINES PLACEES DANS UNECASE SITUEE
 DE VOTRE COTE."
3390 PRINT "CES GRAINES SONT DISTRIBUEES
UNE PAR UNE DANS LES CASES, DANS LE SENS
 INVERSE DES"
3400 PRINT "AIGUILLES D'UNE MONTRE.
SI LA DERNIERE GRAINE DE    VOTRE DEPLAC
EMENT EST PLACEE"
3410 PRINT "DANS VOTRE CASE PERSONNELLE,
VOUS PRENEZ UN SECOND TOUR.": : :
3420 GOSUB 16000
3550 CALL CHAR(96,"")
3600 CALL CHAR(97,"000000000004")
3650 CALL CHAR(98,"000000040004")
3700 CALL CHAR(99,"000040004004")
3750 CALL CHAR(100,"4000400040004")
3800 CALL CHAR(101,"4000400040005")
3850 CALL CHAR(102,"4000400050005")
3900 CALL CHAR(103,"4000500050005")
3950 CALL CHAR(104,"5000500050005")
4000 CALL CHAR(105,"50005000500054")
4050 CALL CHAR(106,"50005000540054")
4100 CALL CHAR(107,"50005400540054")
4150 CALL CHAR(108,"54005400540054")
4200 CALL CHAR(109,"54005400540055")
4250 CALL CHAR(110,"54005400550055")
4300 CALL CHAR(111,"54005500550055")
4350 CALL CHAR(112,"55005500550055")
4400 CALL COLOR(14,5,16)
4450 CALL COLOR(15,9,16)
4500 CALL CHAR(136,"183C7EFF18181818")
4550 CALL CHAR(144,"18181818FF7E3C18")
4600 PRINT "SI LA DERNIERE GRAINE EST
PLACEE DANS UNE CASE VIDE,  DIRECTEMENT
OPPOSEE A UNE"
4610 PRINT "CASE OCCUPEE, TOUTES LES
GRAINES PLACEES DANS CETTE  CASE PLUS LA
 DERNIERE GRAINE"
4620 PRINT "SONT DEPLACEES VERS LA CASE
PERSONNELLE DU JOUEUR.      QUAND UN COT
E EST VIDE, LA"
4630 PRINT "PARTIE S'ARRETE. LE JOUEUR
QUI A LE PLUS DE GRAINES    EST LE VAINQ
UEUR."
4640 PRINT :"UTILISEZ LES FLECHES POUR
 INDIQUER LA CASE ET APPUYER SUR <ENTER>
 POUR VALIDER."
4650 PRINT :"VOTRE ADVERSAIRE EST
 'UMBALA KUBUI', LEGENDAIRE  CHAMPION DE
 CENTRE AFRIQUE.": :
4660 GOSUB 16000
4950 U=11
5000 V=19
5050 W=9
5100 Z=9
5150 DATA 0
5200 FOR X=1 TO 1000
5250 NEXT X
5300 DIM B(13),G(13),F(50)
5350 PRINT : :
5400 INPUT "VOTRE NOM SVP ? ":A$
5450 READ N
5500 E=0
5550 CALL CLEAR
5600 FOR I=0 TO 12
5650 B(I)=3
5700 NEXT I
5750 C=0
5800 F(N)=0
5850 B(13)=0
5900 B(6)=0
5950 PRINT "VOULEZ VOUS COMMENCER ?
(O/N)"
5955 CALL KEY(0,TOUCHE,ETAT)
5960 IF (TOUCHE<78)+(TOUCHE>79)THEN 5955
5970 CALL CLEAR
6000 CALL COLOR(3,2,16)
6050 CALL COLOR(4,2,16)
6100 IF TOUCHE=78 THEN 6250
6150 GOSUB 10650
6200 GOTO 6400
6250 GOSUB 10650
6300 GOTO 6700
6400 CALL HCHAR(V,W,136)
6450 GOSUB 8600
6500 IF E=0 THEN 7250
6550 IF M<>H THEN 6650
```

```
6600 GOSUB 8600
6650 IF E=0 THEN 7250
6700 CALL HCHAR(V,W,32)
6750 GOSUB 13050
6800 IF E=0 THEN 7250
6850 IF M<>H THEN 7050
6900 GOSUB 11050
6950 CALL HCHAR(U,Z,32)
7000 GOSUB 13050
7050 IF E=0 THEN 7250
7100 GOSUB 11050
7150 CALL HCHAR(U,Z,32)
7200 GOTO 6400
7250 GOSUB 11050
7300 IF B(6)>B(13)THEN 7650
7350 IF B(13)>B(6)THEN 8200
7400 FOR X=880 TO 110 STEP -55
7450 CALL SOUND(100,X,0)
7500 CALL SOUND(100,110,30)
7550 NEXT X
7600 GOTO 5500
7650 CALL HCHAR(21,25,152)
7700 CALL HCHAR(21,28,152)
7750 CALL HCHAR(20,25,152,4)
7800 CALL HCHAR(22,25,152,4)
7850 CALL SOUND(300,220,0)
7900 FOR X=220 TO 1020 STEP 50
7950 CALL SOUND(20,X,0)
8000 NEXT X
8050 FOR X=1 TO 1000
8100 NEXT X
8150 GOTO 17000
8200 CALL HCHAR(9,25,152)
8250 CALL HCHAR(9,28,152)
8300 CALL HCHAR(8,25,152,4)
8350 CALL HCHAR(10,25,152,4)
8400 GOSUB 2500
8450 FOR X=1 TO 1000
8500 NEXT X
8550 GOTO 17000
8600 CALL KEY(0,K,S)
8650 CALL HCHAR(V,W,136)
8700 IF S=0 THEN 8600
8750 IF K<>68 THEN 9050
8800 CALL HCHAR(V,W,32)
8850 W=W+3
8900 IF W<25 THEN 8600
8950 W=24
9000 GOTO 8600
9050 IF K<>83 THEN 9350
9100 CALL HCHAR(V,W,32)
9150 W=W-3
9200 IF W>8 THEN 8600
9250 W=9
9300 GOTO 8600
9350 M=(W-6)/3
9400 CALL SOUND(100,440,1,550,1)
9450 CALL SOUND(100,550,1,660,1)
9500 M=M-1
9550 IF B(M)=0 THEN 8600
9600 H=6
9650 GOSUB 9750
9700 GOTO 11050
9750 T=M
9800 GOSUB 12300
9850 E=0
9900 IF T<=6 THEN 10000
9950 T=T-7
10000 C=C+1
10050 IF C>=9 THEN 10150
10100 F(N)=F(N)*6+T
10150 FOR I=0 TO 5
10200 IF B(I)<>0 THEN 10350
10250 NEXT I
10300 RETURN
10350 FOR I=7 TO 12
10400 IF B(I)=0 THEN 10550
10450 E=1
10500 RETURN
10550 NEXT I
10600 RETURN
10650 GOSUB 1400
10700 PRINT "            UMBALA KUBUI": : :
10750 PRINT "  MA";TAB(25);"TA   ":"CASE"
;TAB(25);"CASE"
10800 PRINT : : : : : : :
10850 PRINT :"           ";A$: :
10900 FOR X=12 TO 18
10950 CALL HCHAR(X,8,152,18)
11000 NEXT X
11050 FOR I=12 TO 7 STEP -1
11100 CALL HCHAR(13,9+3*(12-I),B(I)+96)
11150 NEXT I
11200 FOR I=0 TO 5
11250 CALL HCHAR(17,9+3*I,B(I)+96)
11300 NEXT I
11350 D1=INT(B(13)/10)
11400 D2=B(13)-10*D1
11450 CALL HCHAR(9,26,D1+48)
11500 CALL HCHAR(9,27,D2+48)
11550 D1=INT(B(6)/10)
11600 D2=B(6)-10*D1
11650 CALL HCHAR(21,26,D1+48)
11700 CALL HCHAR(21,27,D2+48)
11750 IF B(13)>16 THEN 11900
11800 CALL HCHAR(15,5,B(13)+96)
11850 GOTO 12000
11900 CALL HCHAR(15,5,112)
11950 CALL HCHAR(15,6,B(13)+80)
12000 IF B(6)>16 THEN 12150
12050 CALL HCHAR(15,27,B(6)+96)
12100 RETURN
12150 CALL HCHAR(15,27,112)
12200 CALL HCHAR(15,28,B(6)+80)
12250 RETURN
12300 P=B(M)
12350 B(M)=0
12400 FOR P=P TO 1 STEP -1
```

```
12450 M=M+1
12500 IF M<=13 THEN 12600
12550 M=M-14
12600 B(M)=B(M)+1
12650 NEXT P
12700 IF (B(M)=1)+(M<>6)+(M<>13)<>-3 THE
N 12800
12750 IF B(12-M)<>0 THEN 12850
12800 RETURN
12850 B(H)=B(H)+B(12-M)+1
12900 B(M)=0
12950 B(12-M)=0
13000 RETURN
13050 D=-99
13100 H=13
13150 FOR I=0 TO 13
13200 G(I)=B(I)
13250 NEXT I
13300 FOR J=7 TO 12
13350 IF B(J)=0 THEN 14950
13400 Q=0
13450 M=J
13500 GOSUB 12300
13550 FOR I=0 TO 5
13600 IF B(I)=0 THEN 14150
13650 L=B(I)+I
13700 R=0
13750 IF L<=13 THEN 13950
13800 L=L-14
13850 R=1
13900 GOTO 13750
13950 IF (B(L)=0)+(L<>6)+(L<>13)<>-3 THE
N 14050
14000 R=B(12-L)+R
14050 IF R<=Q THEN 14150
14100 Q=R
14150 NEXT I
14200 Q=B(13)-B(6)-Q
14250 IF C>8 THEN 14650
14300 T=J
14350 IF T<=6 THEN 14450
14400 T=T-7
14450 FOR I=0 TO N-1
14500 IF F(N)*6+T<>INT(F(I)/6^(7-C)+.1)T
HEN 14600
14550 Q=Q-2
14600 NEXT I
14650 FOR I=0 TO 13
14700 B(I)=G(I)
14750 NEXT I
14800 IF Q<D THEN 14950
14850 A=J
14900 D=Q
14950 NEXT J
15000 M=A
15050 Z=3*(15-M)
15100 CALL HCHAR(U,Z,144)
15150 CALL SOUND(100,440,1,550,1)
15200 CALL SOUND(100,440,1,330,1)
15250 GOTO 9750
15300 FOR I=0 TO N-1
15350 PRINT B(I)
15400 NEXT I
15450 END
16000 PRINT TAB(5);"APPUYEZ SUR <ENTER>"
:TAB(7);"POUR CONTINUER"
16010 CALL KEY(0,TOUCHE,ETAT)
16020 IF TOUCHE<>13 THEN 16010
16030 CALL CLEAR
16040 RETURN
17000 CALL CLEAR
17005 PRINT "VOULEZ VOUS REJOUER ?
 (O/N)"
17010 CALL KEY(0,TOUCHE,ETAT)
17020 IF (TOUCHE<78)+(TOUCHE>79)THEN 170
10
17030 IF TOUCHE=79 THEN 5500
17040 CALL CLEAR
17050 END
```

Sapin de Noël

Gilbert Arribet

Ce programme graphique et musical de circonstance est écrit en **Basic Etendu**.

L'utilisation du Basic Etendu pour la réalisation d'un programme graphique présente à la fois un avantage et un inconvénient : les lutins (sprites) permettent de "sortir" de la matrice 24*32 et autorisent ainsi un graphisme plus soigné. Par contre, le jeu de caractères est plus limité (les groupes 15 et 16 ne sont pas accessibles). Il fallait donc étudier très soigneusement la question afin de pouvoir "créer" ce sapin avec les caractères disponibles. Pour cela, l'utilisation d'une feuille de papier millimétrée s'imposait. Ce genre de feuille a un gros défaut : elle est divisée en blocs de dix millimètres par dix alors que les caractères du TI sont formés dans une matrice 8*8. La seule solution était de tracer des lignes au stylo pour simuler l'écran de l'ordinateur. Après cela, il ne restait plus qu'à "noircir les cases". La totalité des caractères devant être redéfinie, la réalisation de ce programme ressemblait beaucoup plus à un travail de bénédictin qu'à un travail de programmeur !

Le programme

Les nombreuses remarques insérées devraient permettre une compréhension aisée de ce programme très simple. Il faut tout de même préciser que le programme joue "Mon beau sapin" lorsque vous appuyez sur la touche "M" et s'interrompt si vous pressez la touche "F".

```
100 ! ****************
120 ! * SAPIN DE NOEL *
140 ! ****************
160 ! * BASIC  ETENDU *
180 ! ****************
200 ! *   COPYRIGHT   *
220 ! *    GILBERT    *
240 ! *    ARRIBET    *
260 ! *      ET       *
280 ! *  99 MAGAZINE  *
300 ! ****************
320 !
340 DIM F1(45),D(45),F2(45),F3(45)
360 !
380 ! efface ecran
400 ! couleurs caracteres :
420 ! transpar./transpar.
440 !
460 CALL CLEAR :: CALL SCREEN(5):: FOR B
=1 TO 14 :: CALL COLOR(B,1,1):: NEXT B
480 !
500 ! DATAs pour musique
520 !
540 DATA 233,4,311,3,311,1.3,311,8,349,4
,392,3,392,1.3,392,6,311,2,349,2,392,2,4
15,4
560 DATA 294,4,349,4,311,10,466,2,466,3,
392,1.3,523,6,466,2,466,3,415,1.3,415,6,
415,2
580 DATA 415,3,349,1.3,466,6,415,2,415,3
,392,1.3,392,8,233,4,311,3,311,1.3,311,8
,349,4
600 DATA 392,3,392,1.3,392,6,311,2,349,2
,392,2,415,4,294,4,349,4,311,8
620 !
640 ! DATAs redefinition
660 ! des caracteres
680 !
700 DATA 0,000101030703070F,07070F1F1F3F
7FFF,3F3F7F7FFFFFFFFF
720 DATA 0000000000000103,0F1F3F7FFFFFFF
FF,0F07010103030707,0F0F1F1F3F7F3F1F,030
30F0F1F3FFFFF,0000000000000403
740 DATA 0F1F0F0703010307,0F0703030767EF
FF,000000040E030101,8080C0C0E0F0F8E,F0F0
F0F8F8FCFCFE,00008080C0C0E0F
760 DATA C0C0E0E0F0F0F8F8,F8FCFFFEFCF8FC
FE,008080C0C0E0F0F8,F8F0E0E0E0F0F8FC,FEF
CFEFFFFFFFFFF,F0E0F0F8FCFCFEFF
780 DATA 0080C0E2ECF8F0E,C0F0F8FCF6F3F0F
C,C0C0E0E0F0F8FCFF,1F0703,FFFFFF78,FFFFF
F7F3E1E0C
800 DATA FFC7830301,FFFFFFFFFF793,FFFFFF
F9E08,FFFFFFFFBF0F0703,E88,FF662
820 DATA FFFFFFE7C3C18,FFFFFFFFFFFF7C1,7E
381,FFFFFFFFF7E3C18,FFFFFFFFFFBF1D08,000
000000000408
840 DATA FFFF7F7F7F7F7F7F,FFFFFFFFFFFFFE
FE,3F3F3F3F3F3F3F7F,FEFEFEFFFFFEFEFE,000
00000010301,FFFFFFFFFFFFFF03
860 DATA FEFEFEFEFFFFFFFC,000000000080C,
81422C1,0000011A242020C,0080453A,0000010
2020C102
880 DATA 0101CE3,0109161020204 08,01122C2
04040808,408,804038040201,00000001028448
3
900 DATA 060908B04,C12618,00000003042458
8,0000000865921,00000006192020C,010202D
428
920 DATA 19668,41221C,000080403B04,030C3
0C,0000609008040403,8040202020100C03
940 DATA 0080412224281,0000814A34,000204
0404483,20202010080 8102,000082412A14,000
08C502
960 DATA 000000001824C202,84482402000020
1,8000020C,0,4,00000000000004
980 DATA 0004,0000000002,2020F8202,1054
38FE38541,40E04,A040A
1000 DATA 105438FE38541,60F0F06,2040C0C0
C0402,385CBEBEBEBE5C38,2070F8702,8870707
088
1020 DATA A040A,40E04,8850205088,2020F82
02,2070F8702,1038FE7C386C82
1040 DATA 60F0F06,2070F8702,40E04,105438
FE38541,3078FCFCFC783,805438FE
1060 DATA FE38548,202070F870202,00400000
000004,0002000000002,00000000040E3F7F,00
00000000010B8FE
1080 !
1100 ! DATAs codes des
1120 ! caracteres
1140 !
1160 DATA 32,32,32,32,32,32,32,33,45,32,
32,32,32,32,32,32,32,32,32,32,32,32,3
4
1180 DATA 46,32,32,32,32,32,32,32,32,32,
32,32,32,32,36,35,81,47,32,32,32,32,32,3
2
1200 DATA 32,32,32,32,32,32,33,80,103,48
,32,32,32,32,32,32,32,32,32,32,32,36,37,
104
1220 DATA 112,49,32,32,32,32,32,32,32
,32,32,32,38,96,111,105,86,50,32,32,32,3
2,32
1240 DATA 32,32,32,32,32,39,102,109,85,1
13,51,32,32,32,32,32,32,32,32,32,36,37,8
3,84
1260 DATA 111,106,46,32,32,32,32,32,32,3
2,32,32,33,82,87,114,99,111,52,47,32,32,
32,32
1280 DATA 32,32,32,32,38,98,111,100,111,
107,97,53,32,32,32,32,32,32,32,41,40,112
,106,111
1300 DATA 92,93,94,95,45,32,32,32,32,32,
32,33,88,89,90,91,111,109,113,110,49,32,
32,32
1320 DATA 32,32,41,37,111,111,104,115,10
8,111,102,111,86,54,32,32,32,32,34,80,91
```

```
,88,93,81
1340 DATA 95,82,90,84,111,48,32,32,32,36
,35,111,107,112,111,115,110,111,105,111,
100,46,32,32
1360 DATA 32,38,99,113,110,97,114,111,10
3,112,111,115,113,53,71,32,32,42,111,106
,111,81,82,90
1380 DATA 88,93,86,82,84,91,55,32,44,43,
112,92,84,111,114,111,115,101,111,110,11
2,107,49,32
1400 DATA 33,91,80,113,111,108,111,103,1
11,109,115,92,93,108,52,45,34,109,104,11
1,102,115,110,114
1420 DATA 105,111,81,87,113,91,80,56,57,
58,59,60,61,62,63,111,111,85,69,70,67,66
,101,49
1440 DATA 32,32,32,32,32,32,32,72,73,68,
32,32,32,32,65,64,32,32,32,32,32,32,32,7
4
1460 DATA 75,32,32,32,32,32,32,32,32,32,
32,32,32,32,76,77,78,79,32,32,32,32,32,3
2
1480 !
1500 ! DATAs couleurs
1520 ! caracteres
1540 !
1560 DATA 13,1,13,1,13,1,13,1,13,1,7,1,1
2,13,10,13,3,13,4,13,8,13
1580 !
1600 ! DATAs positions et
1620 ! couleurs sprites
1640 !
1660 DATA 6,47,115,16,6,125,8,36,144,9,5
7,149,10,93,140,12,141,67,14,168,83,15,1
69,154
1680 DATA 16,87,92,5,119,97,6,67,135,7,1
01,174,8,143,113,9,134,150,10,169,189,11
,74,114
1700 DATA 12,42,107,14,111,141,15,135,18
4,16,149,149,6,150,92,7,27,128,8,169,117
1720 DATA 9,127,137,10,97,109,11,152,128
,7,161,121,7,161,129
1740 !
1760 ! annulation du
1780 ! "PRE-SCAN"
1800 !
1820 GOTO 1940 :: CALL CHAR :: CALL HCHA
R :: CALL SPRITE :: CALL KEY :: CALL CHA
RSET :: CALL DELSPRITE :: CALL SOUND ::
BX :: BY :: C :: CO :: COL :: COL1 :: CO
L2 :: K :: LI :: S :: C$ :: !@P-
1840 !
1860 ! initialisation des
1880 ! tableaux pour
1900 ! musique
1920 !
1940 FOR B=0 TO 45 :: READ F1(B),K :: D(
B)=K*150 :: F2(B)=2*F1(B):: F3(B)=F1(B)/
2 :: NEXT B
1960 !
1980 ! redefinition
2000 ! des caracteres
2020 !
2040 FOR B=32 TO 143 :: READ C$ :: CALL
CHAR(B,C$):: NEXT B
2060 !
2080 ! formation du sapin
2100 !
2120 FOR BX=1 TO 24 :: FOR BY=9 TO 24 ::
 READ C :: CALL HCHAR(BX,BY,C):: NEXT BY
 :: NEXT BX
2140 !
2160 ! couleurs ecran
2180 ! et caracteres
2200 !
2220 CALL SCREEN(15):: FOR B=1 TO 11 ::
READ COL1,COL2 :: CALL COLOR(B,COL1,COL2
):: NEXT B
2240 !
2260 ! positions et couleurs
2280 ! des lutins
2300 !
2320 FOR B=1 TO 28 :: READ COL,LI,CO ::
CALL SPRITE(#B,B+115,COL,LI,CO):: NEXT B
2340 !
2360 ! modifications
2380 ! aleatoires des
2400 ! couleurs et
2420 ! controle du clavier
2440 !
2460 CALL KEY(0,K,S):: IF K=77 OR K=109
THEN 2580 ELSE IF K=70 OR K=102 THEN 266
0
2480 RANDOMIZE :: COL=INT(RND*13)+4 :: I
F COL=13 OR COL=15 THEN 2500 ELSE CALL C
OLOR(#INT(RND*26)+1,COL)
2500 CALL COLOR(11,INT(RND*13)+4,13):: G
OTO 2460
2520 !
2540 ! musique
2560 !
2580 FOR B=0 TO 45 :: CALL SOUND(D(B),F1
(B),2,F2(B),6,F3(B),15):: NEXT B :: GOTO
 2460
2600 !
2620 ! arret du programme
2640 !
2660 CALL CLEAR :: CALL DELSPRITE(ALL)::
 CALL CHARSET :: CALL SCREEN(8):: DISPLA
Y AT(1,1):"""99 MAGAZINE""" :: DISPLAY A
T(3,1):"VOUS SOUHAITE"
2680 DISPLAY AT(5,1):"UN JOYEUX NOEL" ::
 DISPLAY AT(7,1):"ET UNE BONNE ANNEE" ::
 DISPLAY AT(11,1):"RENDEZ-VOUS EN 1984..
."
2700 FOR B=0 TO 2000 :: NEXT B :: CALL C
LEAR :: END
```

PARSEC à l'essai

Marianne Sutz

PARSEC est un logiciel de jeu commercialisé par Texas Instruments sous forme de module enfichable. Ce jeu, de type "spatial", n'a pas grand chose à envier aux jeux de cafés et autres estaminets. En effet, les bruitages sont excellents et l'utilisation du mode BIT-MAP (uniquement accessible depuis un programme en assembleur ou en Pascal) a permis la réalisation d'un graphisme très soigné. De plus, ce jeu utilise le synthétiseur vocal. Celui-ci n'est pas indispensable mais augmente sensiblement l'intérêt. Par contre, bien qu'il soit possible d'utiliser PARSEC avec le seul clavier, les joysticks sont pratiquement indispensables.

Plantons le décor

Vous êtes le pilote d'un vaisseau spatial baptisé "PARSEC" (le parsec est une unité de distance équivalente à 3,26 années lumière) et vous survolez une planète étrangère. Le sol, parsemé d'obstacles, défile à une vitesse proportionnelle à celle de votre vaisseau. Le ciel, étoilé, défile environ deux fois moins vite que le sol et donne ainsi une impression de profondeur. Le bas de l'écran est réservé aux indications sur le déroulement du jeu. Pour ceux qui disposent du synthétiseur, une voix féminine (hé oui !), très bien définie, donne quelques indications supplémentaires.

But du jeu

Des vaisseaux étrangers cherchent à vous détruire. Pour vous défendre, vous disposez d'un laser, mais un problème de refroidissement vous empêche de tirer en continu. Une pression trop insistante sur le bouton de tir provoque l'explosion de PARSEC. Le but du jeu est la destruction d'un maximum d'appareils étrangers. Vous disposez, en début de partie, de cinq vaisseaux. Ce nombre est augmenté en fonction du score. La partie se termine lorsque tous vos appareils sont détruits.

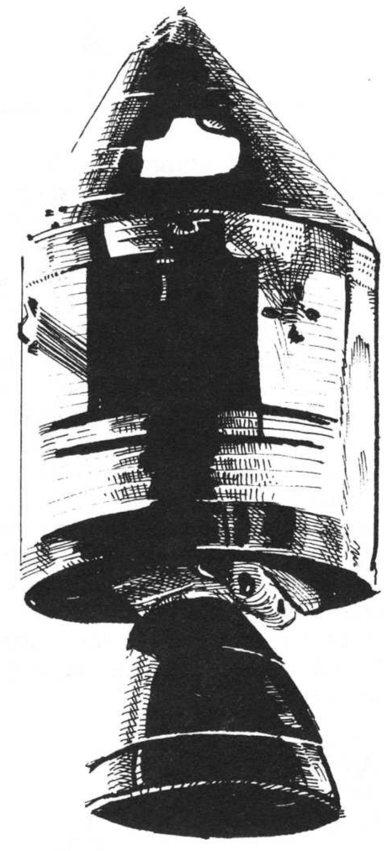
Les adversaires

Plusieurs types de vaisseaux cherchent à vous abattre. Certains essaient de vous percuter alors que d'autres, en plus, disposent de "missiles photons" pour vous détruire.

Les SWOOPERS essaient de vous percuter. Leur vitesse augmente si vous ne réussissez pas à les détruire rapidement.

Les URBITES disposent de deux canons à photons qu'ils utilisent lorsque vous traversez leurs trajectoires.

Les CHASSEURS LFT se comportent de la même façon que les SWOOPERS mais sont plus agressifs.



Les DRATAMITES sont équipés d'un missile et suivent vos déplacements avec une rapidité affolante.

Les SOUCOUPES, redoutables, vous attaquent par derrière et, si elles ne vous touchent pas du premier coup, font demi-tour et essaient de vous percuter par l'avant.

Les BYNITES sont assez lents mais deviennent invisibles dès le deuxième niveau de jeu. Seuls leurs projectiles restent visibles.

Les SATELLITES DESTRUCTEURS apparaissent par surprise à partir du quatrième niveau. Ce sont les ennemis les plus dangereux.

Lorsque vous aurez affronté ces appareils, et avant de changer de niveau de jeu, il vous faudra traverser une barrière d'astéroïdes.

Niveaux de jeu

Le jeu comporte seize niveaux. Au deuxième, vous devrez toucher les vaisseaux deux fois pour les détruire. A partir du troisième niveau, il faudra les toucher 3 fois. Au-delà, ils apparaîtront de plus en plus près de votre appareil. Le fait de changer de niveau augmente aussi la longueur de la barrière d'astéroïdes.

Ravitaillement en carburant

Une bande rouge située en bas de l'écran représente le contenu de votre réservoir de carburant. Il faut faire le plein de temps en temps. Ceci serait très simple si les stations de ravitaillement n'étaient pas placées à l'intérieur de tunnels aux parois irrégulières. Si vous réussissez à sortir du premier tunnel, le prochain ravitaillement se fera dans un tunnel plus long. Il en va de même pour le troisième ; ce troisième tunnel est interminable et il est très difficile d'en sortir indemne.

Si vous tombez en panne de carburant, vous verrez votre vaisseau s'écraser mollement sur le sol de la planète.

Conclusion

PARSEC est certainement le meilleur jeu d'action actuellement commercialisé en France pour le TI-99/4A.

Les bruitages sont plus vrais que nature et le graphisme beaucoup plus sophistiqué que sur les programmes précédents. Il est vrai que ces programmes étaient conçus pour être utilisés sur le TI-99/4, lequel ne comportait pas le mode BIT-MAP. Il faut cependant signaler un défaut : lors des changements de niveau, il arrive qu'un vaisseau explose sans raison apparente. Malgré cela, PARSEC est définitivement à conseiller aux amateurs de jeux "galactiques".

LOGO est-il seulement un langage pour enfants ?

Sophie

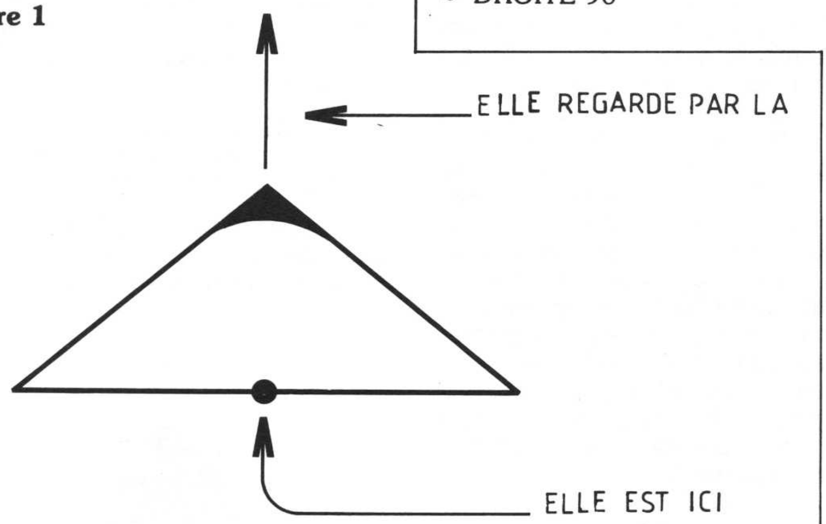
Nous avons jeté, dans l'article précédent, un rapide coup d'oeil sur le monde des lutins. C'est un domaine accessible directement à de jeunes enfants, dès le Cours Préparatoire (5 à 7 ans) mais utilisable aussi bien au-delà de la scolarité primaire : la simulation de mouvements (mouvement uniforme, uniformément accéléré, sinusoïdal, etc...) n'est pas précisément étudiée au niveau de l'école élémentaire...

Aujourd'hui, nous vous proposons une petite excursion dans le domaine de la très célèbre tortue. Prévoyez de bonnes chaussures, car, si nous commençons à un niveau très simple, nous vous emmènerons voir des exemples un peu plus retors !

La tortue a commencé à faire partie de LOGO (et a été reprise ensuite par d'autres langages, comme Pascal) au début des années 1970, c'est-à-dire quand LOGO avait déjà à peu près cinq ans d'existence. Auparavant, on traitait surtout deux domaines plus traditionnels, dont nous n'avons pas encore parlé en détail, les nombres et le texte. Les premières tortues étaient des robots relativement encombrants, d'environ deux pieds - pardon, soixante centimètres - de diamètre, munis de deux roues, d'un phare que l'on pouvait allumer ou éteindre, d'un avertisseur sonore que l'on pouvait actionner, d'un traceur que l'on pouvait mettre en position haute (quand on s'intéressait principalement aux déplacements) ou basse (si l'on réalisait des desins) et de divers capteurs, en particulier des palpeurs capables de renvoyer à l'ordinateur de l'information quand la tortue rencontrait un obstacle. Tous les "organes" de ce robot étant protégés par une coupole hémisphérique en plastique transparent, cet objet baladeur qui se déplaçait lentement au sol pouvait raisonnablement se voir dénommé tortue. Certaines races de tortues étaient reliées par un câble à l'ordinateur, comme celle du Groupe LOGO du M.I.T. ; d'autres, comme la tortue baptisé Irving, chez Bolt, Beranek et Newman, étaient télécommandées.

Depuis, la technique ayant quelque peu évolué, on dispose de tortues d'écran (par opposition aux anciennes tortues de sol), qui apparaissent simplement comme un petit triangle sur l'écran du moniteur.
La tortue est, si vous voulez bien me permettre un raccourci audacieux, un point qui regarde. Autrement dit, elle se trouve ici, en ayant le nez, ou le regard, dirigé par là. C'est pour cette raison que les tortues d'écran sont schématisées par un triangle.

Figure 1



Cette tortue se commande à l'aide d'ordres extrêmement simples : AVANCE et RECULE pour la faire changer d'emplacement (il faut lui préciser le nombre de pas dont elle doit se déplacer), DROITE et GAUCHE pour la faire pivoter dans le sens des aiguilles d'une montre, ou à l'inverse dans le sens trigonométrique (il faut lui indiquer le nombre de degrés dont elle doit tourner). A l'aide de ces seuls quatre ordres, on va pouvoir entreprendre quelques dessins. Nous introduirons de nouveaux primitifs (termes déjà connus de l'ordinateur) au fur et à mesure des besoins.
De même qu'on appelle les lutins, il faut tout d'abord appeler la tortue. La séquence :

- APPELLE TORTUE
- AVANCE 40
- DROITE 90
- AVANCE 60

réalise un L à l'envers:

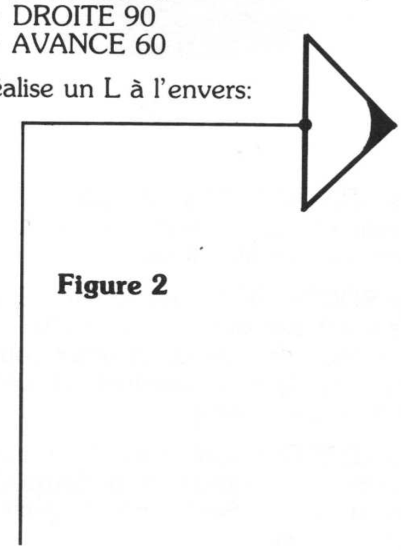


Figure 2

Essayons de terminer ce rectangle. Ne lui proposez pas :
- DESCENDS 40

sinon elle vous répondrait :
- APPRENDS-MOI-DESCENDS

Manifestement, il faut la faire tourner, ici. En se tortillant au besoin un peu sur son siège pour s'identifier à la tortue, n'importe quel enfant finira par trouver :
- DROITE 90

Elle regarde maintenant vers le bas. Il ne nous reste plus qu'à lui faire tracer un L "vu dans un miroir", en la faisant redescendre puis rejoindre son point de départ au centre de l'écran. Essayons :
- AVANCE 40 GAUCHE 90 AVANCE 60

Hé bien ! C'est raté, car elle a tourné vers sa gauche qui, puisqu'elle avait la tête vers le bas, est à votre droite.

Soit on efface tout et on recommence, en changeant le dernier GAUCHE 90 en DROITE 90, soit on pense qu'on est plus tout à fait à l'époque de Jules Ferry - même si le traceur de notre tortue continue d'être nommé PLUME - et que nos modernes écoliers évitent les ratures à l'aide d'un feutre effaceur :
- EFFACEUR
- RECULE 40

voilà le trait déplaisant disparu. Donc, il fallait mettre DROITE au lieu de GAUCHE :
- DROITE 90

Mais ceci ne fait que remettre la tortue la tête en bas. Il nous faut encore la faire pivoter :
- DROITE 90 AVANCE 60

Tiens, un côté fantôme. Hé oui, nous avons toujours notre effaceur au lieu d'un traceur. Revenons sur nos pas :
- RECULE 60

puis reprenons de quoi dessiner :
- POSEPLUME
- AVANCE 60

Ouf, ça y est ! Elle m'a dessiné un rectangle. Curieusement, si je lui demande :

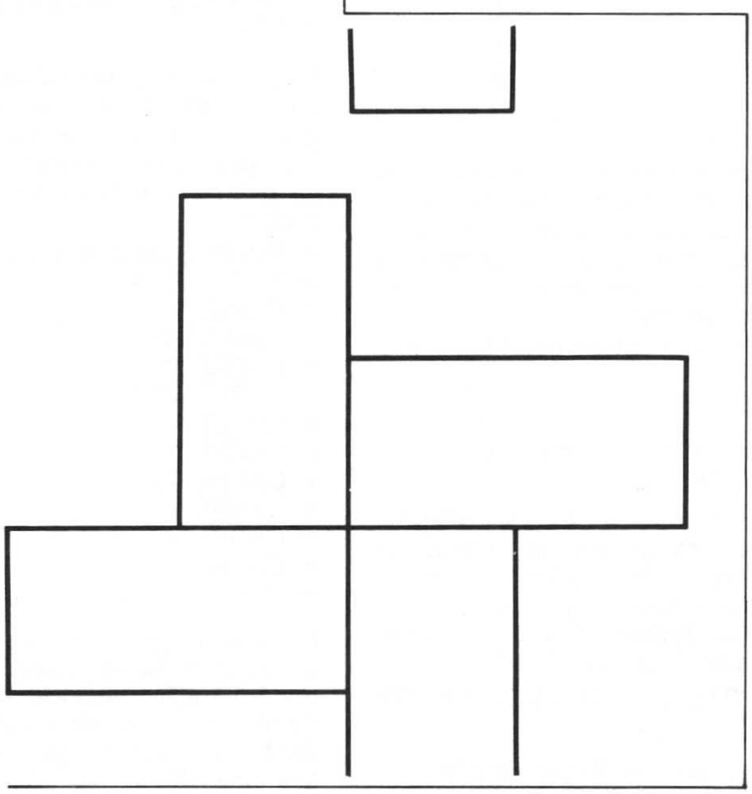
• RECTANGLE
l'ordinateur répond :
• APPRENDS-MOI RECTANGLE
Là encore, c'est très simple :

• POUR RECTANGLE

On passe alors sur une page verte (la tortue est confortablement installée sur une feuille de salade pour écouter ce qu'on lui enseigne ?) et en appuyant sur ENTER on crée une ligne sur laquelle on va pouvoir inscrire les instructions :
• AVANCE 40 DROITE 90
• AVANCE 60 DROITE 90
• AVANCE 40 DROITE 90
• AVANCE 60
FCTN 9 pour quitter la page verte et revenir à l'ancienne page bleue. Le précédent rectangle est toujours là. Tapons :
• RECTANGLE
en voilà un autre à côté, puis :
• RECTANGLE
et un autre au-dessous...

On peut continuer. Mais là, il se passe quelque chose d'inattendu : au lieu d'interférer avec les lignes de texte au bas de la page, un bout du rectangle se trouve reporté en haut de l'écran :

Figure 3



Les enfants expliquent cela très joliment : "elle n'avait plus assez de place en bas, alors elle a continué à descendre en haut".

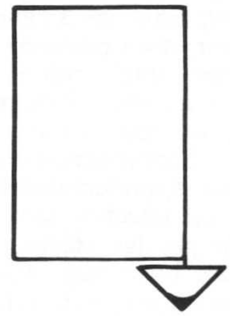
Et si nous lui apprenions à dessiner un triangle, pour construire un moulin à vent à l'aide du rectangle en position "debout" ? Pour nous simplifier l'existence, nous allons lui enseigner un triangle équilatéral ; il faut donc trois côtés de 30 pas, et chacun sait (au moins parmi les adultes) que dans un triangle équilatéral les angles mesurent 60 degrés. Allons-y :
• POUR TRIANGLE
• REPETE 3 [AVANCE 30 DROITE 60]
• FIN
(FIN est déjà écrit sur la page éditeur, inutile de le retaper, sinon l'ordinateur va protester, par un "FIN DE QUOI ?" péremptoire).

Pensons maintenant aux problèmes de positionnement : nous voulons un rectangle debout et non couché, donc il faut commencer (après avoir obtenu une page propre grâce à EFFACEECRAN - attention, en un seul mot -) par tourner la tortue :

• GAUCHE 90
• RECTANGLE

Voilà les murs

Figure 4



Il nous faut alors reparcourir la moitié du rectangle, sinon notre toit va s'enfoncer dans le sol !

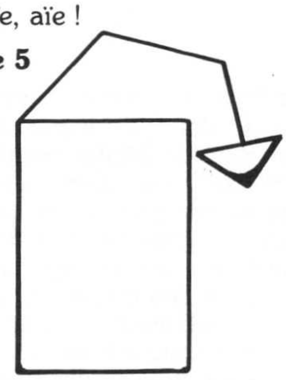
• DROITE 90 AVANCE 40
• DROITE 90 AVANCE 60

Attention ! La tortue regarde maintenant vers le Nord (ou vers le plafond, selon le système de référence auquel vous vous raccrochez !) : il faut d'abord l'incliner un peu, pour la mettre dans la direction de la pente du toit. De combien ? 60 ? Cela paraît beaucoup ! Si on essayait un peu moins, 40 par exemple.

• DROITE 40
• TRIANGLE

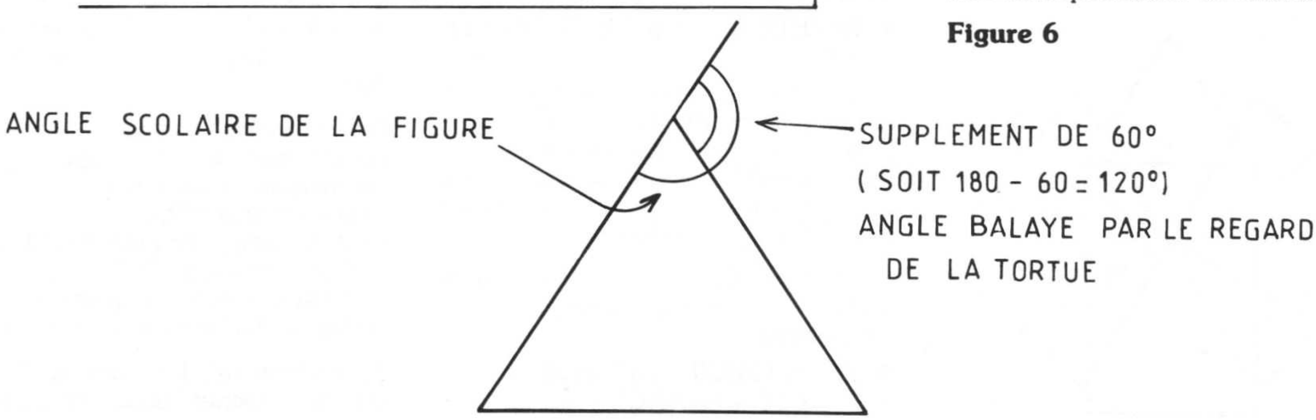
Aïe, aïe, aïe !

Figure 5



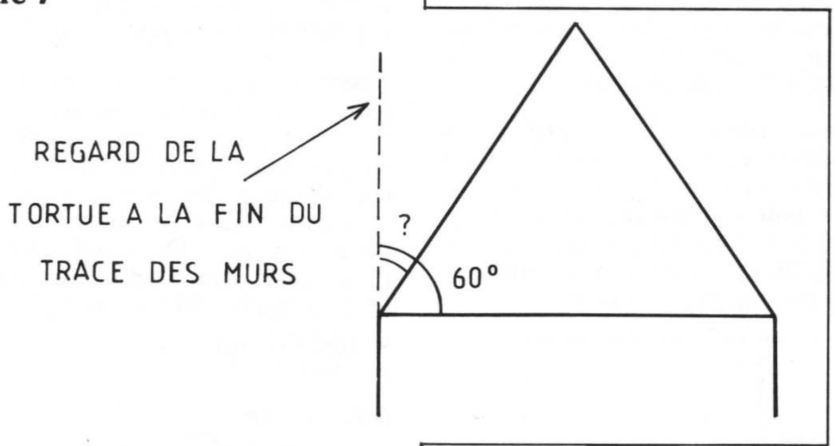
Ce n'est pas un toit, ça !
Que s'est-il passé ? Tout simplement, nous avons appliqué machinalement de vieilles connaissances, sans nous assurer qu'elles étaient pertinentes dans la situation présente. Car la tortue n'a que faire de l'angle scolaire de la figure, ce qui l'intéresse, c'est l'angle que balaie son regard (pour reprendre certaine terminologie familière aux spécialistes de cinématique).

Figure 6



Nous allons bien faire appel à nos souvenirs de géométrie élémentaire (ou construire une connaissance équivalente sur les angles supplémentaires), mais pas à l'aveuglette Et maintenant, nous pouvons même répondre avec précision à notre recherche de l'orientation de la tortue pour la pente du toit, au lieu de nous contenter d'approximations. La méthode des approximations successives, qui est effectivement beaucoup employée par les élèves, est excellente, mais beaucoup plus coûteuse en temps : c'est la raison pour laquelle, par souci d'économie, il va devenir plus payant de raisonner que d'essayer et recommencer.

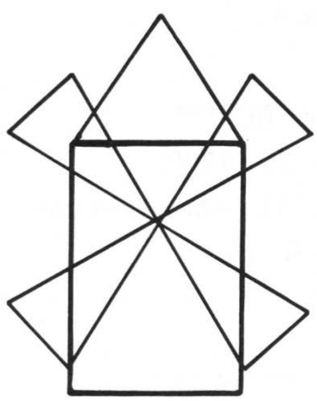
Figure 7



L'angle "inconnu" est le complémentaire de 60, donc 30 degrés. Je vous laisse le soin de piloter votre tortue pour réaliser les ailes. Je vous conseille de commencer par une aile, que vous baptiserez par exemple AILE quand vous enseignerez à la machine à se débrouiller toute seule, puis de coordonner les quatre ailes avec des rotations adéquates - là encore, on va retrouver des calculs sur les angles - en une procédure baptisée 4 AILES par exemple. Il ne vous restera plus alors qu'à lui apprendre

- POUR MOULIN
- RECTANGLE
- TRIANGLE
- 4AILES
- FIN

Figure 8



Voilà un projet graphique où l'objectif de l'élève est de se rendre maître de la tortue et de lui faire réaliser un dessin, en général figuratif. Après cet exemple qui fait travailler les connaissances sur ordre de grandeur, mesure de longueurs et d'angles, notions de symétries, etc..., voyons un projet géométrique. Là, le but que s'assigne, en général implicitement, l'enfant n'est pas seulement le tracé d'une figure, mais bien plutôt la compréhension des propriétés caractéristiques de l'objet géométrique considéré.

Commençons par un projet simple : un carré. La première description du carré que l'on peut constater à travers les réalisations d'enfants avec LOGO est "c'est un machin qui est quatre fois pareil". C'est pourquoi on trouvera beaucoup d'exemples, y compris chez des élèves de niveau collège, tels que :

- REPETE 4 [AVANCE 50 DROITE 50]

ou :

- REPETE 4 [AVANCE 100 DROITE 100]

qui vont amener à une réflexion complémentaire sur la notion de translation et de rotation, et accroître le bagage intuitif sur l'angle.
On arrive finalement à la description d'un carré de 30 pas de côté :

- REPETE 4 [AVANCE 30 DROITE 90]

ou d'un carré de 70 pas de côté :

- REPETE 4 [AVANCE 70 DROITE 90]

et immanquablement, au bout d'un temps plus ou moins long, s'exprime l'intérêt pour un moyen de ne pas fixer d'emblée cette longueur du côté, mais de se réserver le choix au moment de l'utilisation.

La notion de paramètre est alors mère, et l'on arrive à une procédure paramétrée :

- POUR CARRE : LONGUEUR
- REPETE 4 AVANCE : LONGUEUR DROITE 90]
- FIN

A ce moment de l'approche du concept, la syntaxe ":" est uniquement justifiée comme discriminateur entre un terme qui indique une action à entreprendre (AVANCE, CARRE), et un terme qui désigne une information.

- CARRE 30

dessine un carré de trente pas de côté et :

- CARRE 70

en trace un de 70 pas.
Comme précédemment avec le RECTANGLE, le TRIANGLE, et 4AILES, cette nouvelle procédure peut être utilisée dans d'autres procédures :

- POUR ECHELLE : TAILLE
- REPETE 5 [CARRE : TAILLE AVANCE : TAILLE]
- FIN

Ici, on a à nouveau une procédure paramétrée. On peut effectuer des manipulations sur les paramètres :

- POUR BRACELET : TAILLE
- REPETE 10 [CARRE : TAILLE AVANCE : TAILLE/2]
- FIN

Le paramètre peut, au choix de l'auteur, porter ou non le même nom que celui de la procédure CARRE. Une procédure paramétrée peut servir à créer une procédure non paramétrée :

- POUR CARRE.A.MALICES
- INVERSEUR
- CARRE 10
- CARRE 20
- CARRE 30
- CARRE 40
- CARRE 50
- CARRE 60
- CARRE 70
- CARRE 80
- CARRE 90
- CARRE 100
- FIN

Pouvons-nous nous servir de cet acquis pour dessiner une spirale carrée ? Il s'agit d'avancer d'un certain nombre de pas, de pivoter à angle droit, d'avancer un peu plus que la fois précédente, de pivoter à nouveau à angle droit, et de recommencer cette séquence de deux instructions.

On trouve ici un schéma de pensée extrêmement familier, depuis
"la meilleur façon d'marcher
c'est encore la nôtre
c'est de mettre un pied d'vant l'autre
et d'recommencer."
jusqu'aux motifs jacquard en tricot ou les points fantaisie au crochet.

Si, en général, les patrons de tricot ou de crochet vous indiquent le

nombre de fois qu'il convient de répéter le motif, dans la première circonstance - la marche - on le prévoit rarement à l'avance ! On a plutôt tendance, avant de recommencer, à vérifier qu'on n'a pas encore atteint le but de la promenade, et, si l'on est arrivé à destination, on s'arrête. C'est exactement ce que nous allons faire pour la spirale; nous nous arrêterons quand le côté sera devenu grand, disons supérieur à 100 :

- POUR SPIRALE.CARRE : COTE
- AVANCE : COTE DROITE 90
- SI : COTE>100 ALORS STOP
- SPIRALE.CARRE : CO+10
- FIN

L'utilisation de la procédure elle-même au cours de sa propre définition s'appelle récursion.

Si l'on ne veut pas fixer à l'avance la quantité dont on augmente la longueur du côté à chaque étape, on peut se constituer une procédure dépendant de deux paramètres, le côté et l'augmentation :

- POUR SPICAR : COTE : AUGMENTATION
- AVANCE : COTE DROITE 90
- SI : COTE>100 ALORS STOP
- SPICAR : COTE+ : AUGMENTATION : AUGMENTATION
- FIN

Si l'on souhaite une spirale pas nécessairement carrée, rien de plus facile : on introduit un troisième paramètre, l'angle dont on tourne à chaque étape.

- POUR SPIRALE : COTE : PLUS : ANGLE
- AVANCE : COTE DROITE : ANGLE
- SI : COTE>100 ALORS STOP
- SPIRALE : COTE+ : PLUS : PLUS : ANGLE
- FIN

L'ordre dans lequel sont annoncés les paramètres sur la ligne de titre n'est pas nécessairement celui dans lequel ils sont utilisés au cours de la procédure. Rien ne vous empêche de modifier la façon dont les paramètres interviennent lors des appels successifs :

- POUR ECHANGE : COTE : ANGLE
- AVANCE : COTE DROITE : ANGLE
- SI : COTE>100 ALORS STOP
- ECHANGE : ANGLE : COTE
- FIN

ou :

- POUR FOLKLO : COTE : PLUS : ANGLE
- AVANCE : COTE DROITE : ANGLE
- SI : COTE>100 ALORS STOP
- FOLKLO : COTE+ : PLUS : PLUS★2 : ANGLE− : PLUS
- FIN

On pourra se poser des questions sur l'utilité du test dans ECHANGE (l'arrêt de panique s'obtient en appuyant sur FCNT 9 !). On commence à se faire une légère idée de la foule de possibilités qu'offre un espace de paramètres de dimension 3... Cette souplesse et cette richesse incite déjà à penser que la récursion est autre chose qu'une simple boucle FOR-NEXT.

Essayons de regarder cela de plus près. Reprenons la première spirale carrée, à laquelle nous ajoutons une instruction pour afficher la longueur du côté (pour voir toutes les valeurs, on ne va pas à la ligne après l'affichage, mais on reste sur la même ligne ; pour séparer deux valeurs consécutives, on fait afficher, toujours en restant sur la même ligne, des points de suspension).

- POUR SPI : COTE
- AVANCE : COTE DROITE 90
- AFR : COTE AFR "...
- SI : COTE>100 ALORS STOP
- SPI : COTE+10
- FIN

Quand on tape :

- SPI 5

la spirale carrée se dessine, et au fur et à mesure s'inscrivent les valeurs du côté :

5...15...25...35...45...55
...65...75...85...95...105...

Bon, rien de très surprenant jusque là.

Essayons cette autre procédure :

- POUR IPS : COTE
- AVANCE : COTE DROITE 90
- SI : COTE>100 ALORS STOP
- IPS : COTE+10
- AFR : COTE AFR "...
- FIN

Avant de regarder la suite, voudriez-vous avoir la gentillesse d'essayer d'imaginer ce qui va se passer avec IPS ? Attention, défense de tricher : on n'a pas le droit de chercher tout de suite la réponse, ni dans la revue, ni en faisant exécuter la procédure IPS par l'ordinateur !!!

Alors, qu'aviez-vous prévu ? Voici quelques suggestions que nous avons déjà entendues :

a) la procédure n'affiche rien

b) la procédure n'affiche qu'une seule valeur (selon les cas, on propose 5 ou 105)

c) la procédure fait la même chose que SPI

d) la procédure affiche la longueur totale de la spirale.

Si vous avez d'autres propositions, je suis preneur !

Essayez maintenant IPS 5 sur votre ordinateur. La tortue commence par dessiner la spirale, mais sans rien tracer. Les tenants de la solution (a) commencent à se dire : tiens, tiens, j'avais donc bien deviné ? Hun paquet d'affichage :

95...85...75...65...55
...45...35...25...15...5

Mais pourquoi diable est-ce à l'envers ?

Regardons cela de plus près. Quand on tape IPS 5, la procédure IPS commence, avec la valeur 5 pour : COTE. Elle fait avancer la tortue de 5 pas, et la fait tourner de 90 degrés vers la droite. Comme 5 est bien loin d'être supérieur à 100, on ignore le STOP, et alors on a besoin que s'exécute IPS 15. Mais IPS 5 N'a pas terminé son travail ! Cette procédure se met en quelque sorte en attente, et appelle :

- IPS 15.

Cette procédure (une copie de IPS) commence, avec la valeur 15 pour : COTE. A son tour, elle fait bouger la tortue, puis se trouve arrêtée en plein élan, et appelle un autre exemplaire de cette même procédure.

Sautons quelques étapes, et regardons ce qui se passe avec 85 comme valeur pour : COTE.

- IPS commence avec la valeur 85 pour : COTE. Là encore, après avoir fait bouger la tortue, elle ignore le STOP, et appelle une nouvelle version de IPS ;
- IPS commence avec la valeur 95 pour : COTE, bouge toujours la tortue, 95 n'est pas encore une valeur assez grande pour que le test d'arrêt s'applique. On a donc recours à un nouveau IPS.

Cet IPS commence, avec la valeur 105 pour : COTE. La tortue avance et pivote, comme elle l'avait fait lors de chacun des précédents appels. Mais cette fois-ci, 105 étant supérieur à 100, la procédure a terminé son travail et "rend son tablier" à son employeur, c'est-à-dire retourne le contrôle à IPS 95.

Or IPS 95 a encore une ligne d'instructions à exécuter, l'affichage de sa propre valeur de : COTE, soit 95. Puis, à son tour, elle rend le contrôle à la procédure qui l'a appelée... et ainsi de suite jusqu'à la fin de l'exécution de la procédure IPS 5.

Autant la récursion terminale (celle de SPI et des exemples antérieurs) où l'appel à la procédure se fait sur la dernière ligne d'instructions est relativement facile à interpréter, autant la récursion non terminale paraît, aux premiers essais, tout-à-fait déroutante.

C'est pourtant un outil extrêmement puissant, dont disposent tous les langages de haut niveau récents.

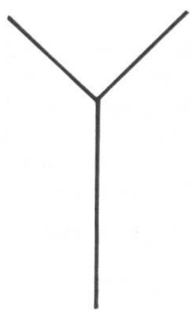
Donnons deux exemples, dans le cas de la tortue toujours, (mais ceci peut être appliqué dans les autres domaines LOGO, bien sûr).

Tout d'abord un arbre binaire :
au niveau 1, c'est seulement un tronc.

Figure 9

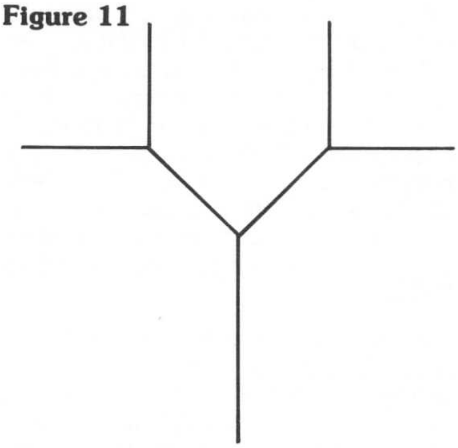
Au niveau 2, le tronc a deux branches.

Figure 10



Au niveau 3, chaque branche se divise en deux.

Figure 11



On passe du niveau n au niveau n+1 en subdivisant en deux chaque rameau terminal. Nous allons raccourcir les branches à chaque niveau, pour faire plus nature et également pour éviter d'envahir trop rapidement tout l'écran.

Regardons ce qui se passe à la première division en deux branches; on remarquera que c'est exactement le même processus qui se répète ensuite :

- POUR BRANCHE : LONGUEUR

Il faut d'abord tracer la branche principale (au premier coup, c'est le tronc) :

- AVANCE : LONGUEUR

puis tourner de l'angle adéquat pour amorcer la branche gauche :

- GAUCHE 45

et la tracer :

- BRANCHE : LONGUEUR−5

ensuite pivoter pour se mettre dans la direction où va pousser la deuxième branche :

- DROITE 90

tracer la deuxième branche :

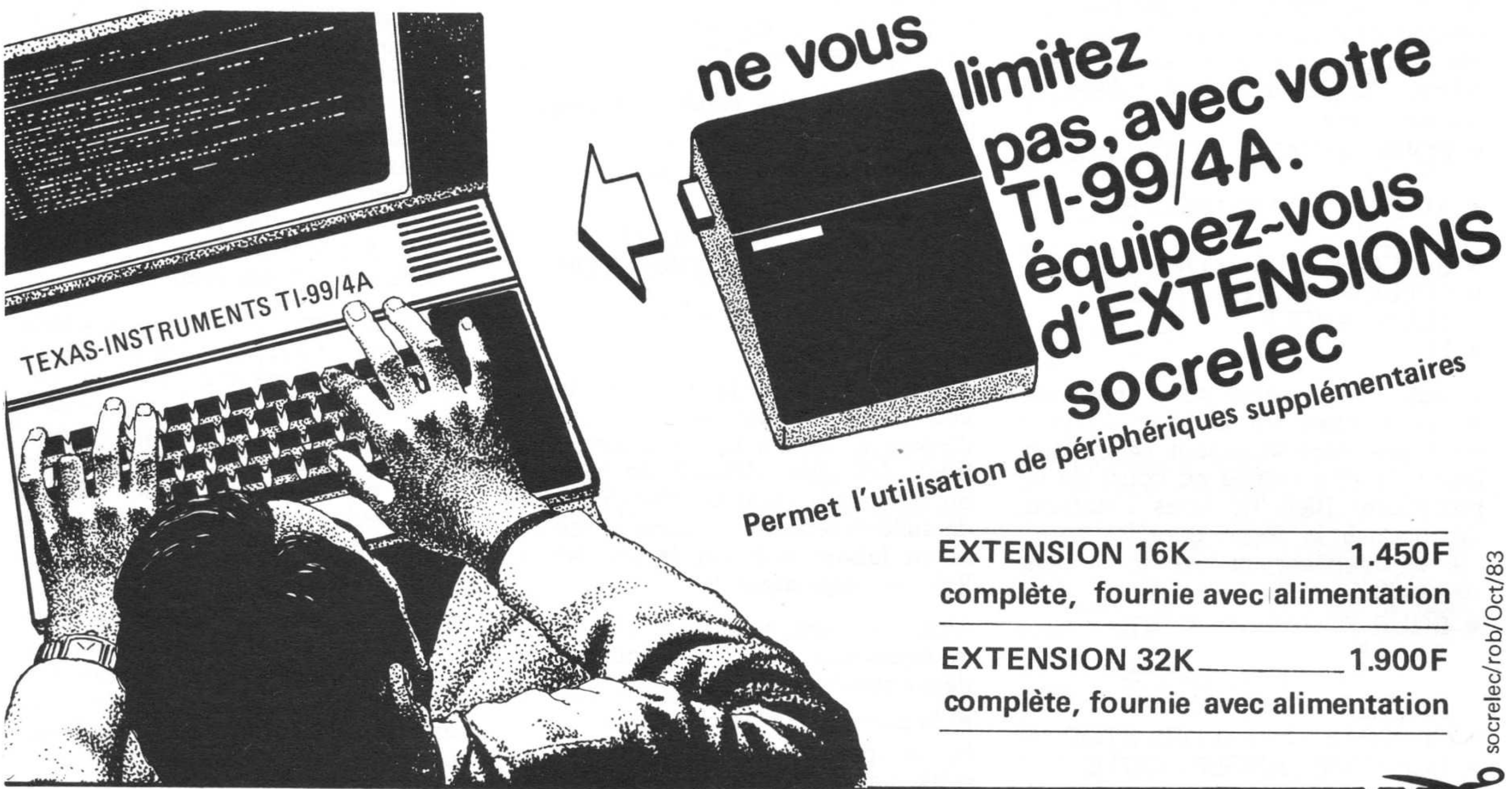
- BRANCHE : LONGUEUR−5

Mais, comme le processus est symétrique, on peut s'attendre à avoir une procédure symétrique. Une autre raison est qu'ici on est resté en bout de branche, et qu'au niveau suivant on va donc rallonger cette branche :

- GAUCHE 45
- RECULE : LONGUEUR

Il faut maintenant penser qu'au niveau 0 le processus doit s'arrêter : on arrive alors à la procédure que nous allons rebaptiser :

- POUR ARBRE : LONGUEUR : NIVEAU
- SI : NIVEAU=0 ALORS STOP
- AVANCE : LONGUEUR
- GAUCHE 45
- ARBRE : LONGUEUR−5 : NIVEAU−1
- DROITE 90



EXTENSION 16K ______ 1.450F
complète, fournie avec alimentation

EXTENSION 32K ______ 1.900F
complète, fournie avec alimentation

socrelec/rob/Oct/83

REVENDEUR A PARIS : **MVI** 27, Rue Vaneau 75007 PARIS - Tél: 551 66.77

BON DE COMMANDE à SOCRELEC
26, rue du Pavé des Gardes 92370 CHAVILLE - Tél: (1) 750 53.56

Nom ____________ Adresse ____________

Prénom ____________ ☐☐☐☐☐ Ville ____________

TOUS NOS PRIX SONT T.T.C.

Nous encaissons votre chèque, à l'expédition de votre commande.

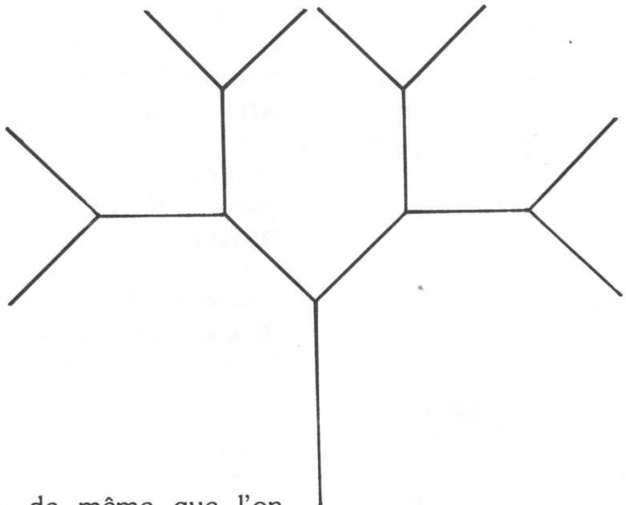
Signature des parents obligatoire pour les mineurs.

	Quantité
Extension 16K	
Extension 32K	

- ARBRE : LONGUEUR−5 : NIVEAU−1
- GAUCHE 45
- RECULE : LONGUEUR
- FIN

Cette procédure utilise donc une double récursion non terminale. Voici l'effet obtenu au niveau 4, toujours avec cette "simple" procédure de dix lignes :

Figure 12



Bien entendu, de même que l'on peut introduire deux appels récursifs (dont un au moins est nécessairement non-terminal !), on peut s'offrir le luxe de procédures un peu plus riches, quoique toujours extrêmement courtes.

Par exemple, la célèbre courbe en flocon de neige se réalise au moyen de deux procédures :

- POUR FLOCON : TAILLE : PROFONDEUR
- REPETE 3 [COTE : TAILLE : PROFONDEUR DROITE 120]
- FIN

et la procédure COTE qui utilise quatre appels récursifs :

- POUR COTE : TAILLE : PROFONDEUR
- SI : PROFONDEUR=0 ALORS AVANCE : LONGUEUR STOP
- COTE : LONGUEUR/3 : PROFONDEUR−1
- DROITE 120

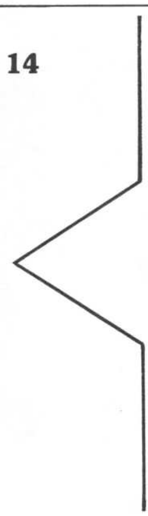
Figure 13

- COTE : LONGUEUR/3 : PROFONDEUR−1
- GAUCHE 60
- COTE : LONGUEUR/3 : PROFONDEUR−1
- FIN

Au niveau 0, COTE trace tout simplement un segment :

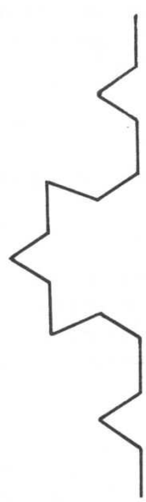
Au niveau 1, c'est un segment comportant une dent :

Figure 14



Au niveau 2, un dessin plus dentelé encore :

Figure 15



Remarquons qu'un paramètre et une procédure peuvent avoir le même nom (par exemple COTE), sans que cela perturbe en rien l'ordinateur. Il vaut mieux l'éviter, le cerveau humain ayant, lui, tendence à s'embrouiller).

Vous retrouverez ces deux exemples, et bien d'autres, faisant appel à la récursion, dans Turtle Geometry, d'Abelson et Di Sessa, M.I.T. Press, 1981. Attention, ce n'est pas un livre pour l'école primaire : on y traite de sujets du niveau maîtrise de maths ou de physique, comme les géométries non euclidiennes, la topologie algébrique, la relativité généralisée...

Outre ces capacités récursives, qui - il faut bien le reconnaître - sont un peu déroutantes au début (mais peut-être pas plus que l'incongru A=A+1 en Basic), la tortue LOGO peut s'adapter à la géométrie cartésienne la plus traditionnelle. On peut avoir un certain nombre de renseignement sur l'état de la tortue, en particulier son abscisse :

- AFFICHE XCOR

son ordonnée

- AFFICHE YCOR

ainsi que son cap, c'est-à-dire l'angle de son regard au moment présent par rapport à sa direction initiale quand on à fait apparaître la tortue sur l'écran, compté positivement dans le sens des aiguilles d'une montre et mesuré en degrés :

- AFFICHE CAP

On peut aussi positionner la tortue directement en un endroit dont on connait les coordonnées avec :

- FIXEXY abscisse ordonnée

ou lui imposer de regarder dans une direction donnée avec :

- FIXECAP valeur

Voilà, je pense, qui vous aura convaincu que la réponse à la question posée par le titre est bien évidemment négative.

Si vous voulez m'écrire, vous pouvez m'adresser du courrier au journal, ou, de préférence, à GREPACIFIC, 51 Boulevard des Batignolles, 75008 PARIS. Je vous rappelle que nous utilisons le LOGO sur le TI-99/4A (ainsi que plusieurs autres versions de LOGO) mais que nous ne connaissons pas beaucoup les autres ressources de cette excellente machine - nous avons déjà fort largement à faire pour tenter d'exploiter toutes les richesses du TI-LOGO. N'essayez pas de venir nous voir à l'adresse indiquée, c'est seulement notre boîte aux lettres : nous travaillons dans diverses écoles, dans des collèges, en Ecoles Normales, en Universités, et dans d'autres conditions encore... et le terrain nous absorbe beaucoup.

Le Mastermind en Assembleur

Jean-Luc Bazanegue

Ces derniers temps, nous avons reçu un grand nombre de lettres de lecteurs nous demandant de publier des programmes en assembleur. Comme vos désirs sont des ordres, nous vous proposons cette fois un Master Mind écrit en ce langage.

Ce programme met en évidence la supériorité de l'assembleur par rapport au Basic. Deux modes d'affichage sont utilisés : le mode texte et le mode Bit-Map.

Le mode texte permet d'obtenir un affichage sur 40 colonnes. Le mode Bit-Map autorise un affichage point par point, il est donc possible d'adresser indépendamment les 49152 points (ou pixels) constituant l'image générée par le TI-99/4A. Avec ce mode, deux couleurs par ligne de 8 pixels peuvent être choisies (couleur des points "on" et couleur des points "off".)

Particularité du jeu

Le jeu consiste à découvrir, en dix essais maximum, un code constitué par des couleurs choisies par l'ordinateur. Ces couleurs sont réparties dans cinq cases. Le nombre de couleurs peut varier de deux à dix : cela constitue les niveaux de jeu.

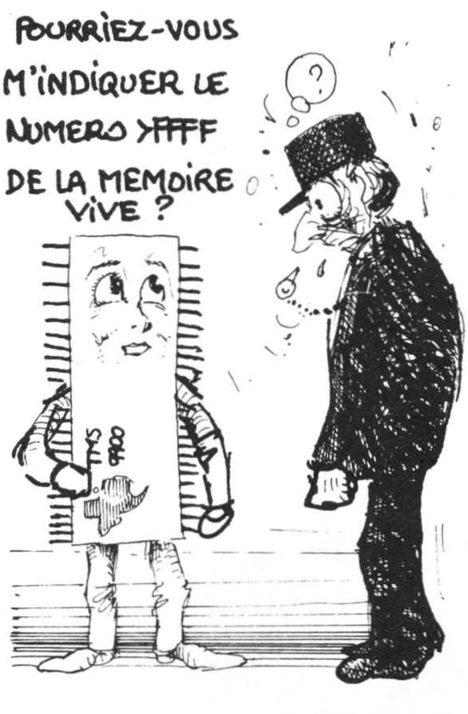
A chaque essai, le TI vous indiquera le nombre de couleurs bien et mal placées, ainsi que vos erreurs de logiques.

Mode d'emploi du programme

Pour utiliser ce programme, il faut posséder une extension 32K et une unité de disquettes. Le code objet doit être chargé en mémoire par un programme en Basic Etendu ou en Basic TI. La routine suivante peut être utilisée :

10 CALL INIT
20 CALL LOAD ("DSK1.MASTER")
30 CALL LINK ("MASTER")

Des indications insérées dans le listing permettent de modifier le programme pour qu'il puisse être chargé par l'option LOAD AND RUN de l'éditeur/assembleur ou de la mini-mémoire.



Touches utilisées

1 à 0 : couleurs à placer dans une case. Un tableau placé sur la droite de l'écran donne les correspondances couleurs/numéros de couleurs.

ENTER : validation de l'essai. Si les cinq cases ne sont pas occupées, la validation est impossible.

Flèche à gauche (S) : déplacement du cadre repère vers la gauche.

Flèche à droite (D) : déplacement du cadre repère vers la droite.

E : effacement d'une couleur et déplacement du cadre vers la droite.

AID : abandon. La solution est affichée.

QUIT : permet de sortir du programme. Retour à la mire.

REDO : en fin de jeu, reprise d'une nouvelle partie de même niveau (même nombre de couleurs).

BACK : en fin de jeu, retourne à l'image d'introduction. Cela permet de choisir un niveau de jeu différent.

Sur l'écran, les chiffres situés dans la colonne de gauche indiquent les couleurs bien placées, et les chiffres dans la colonne de droite indiquent les couleurs mal placées.

Lors d'une erreur de logique, une flèche vers la gauche indique une erreur par rapport aux couleurs bien placées dans les essais précédents. Une flèche vers la droite indique une erreur par rapport aux couleurs mal placées.

Dans le listing, vous trouverez de nombreuses routines très facilement implantables dans vos propres programmes en assembleur : initialisation de la mémoire vidéo pour le mode Bit-Map, génération de nombres aléatoires, affichage d'un point sur l'écran, etc...

Le listing est relativement long et vous ne trouverez peut-être pas le temps d'effectuer la saisie. Dans ce cas, nous pouvons vous envoyer, contre 50 FF, une disquette contenant les sept fichiers source et le fichier objet.

```
*****************************************************************
*              *                                                *
* MASTER MIND  * COPYRIGHT "99 MAGAZINE" ET JEAN-LUC BAZANEGUE  *
*              *                                                *
*****************************************************************
* Ce programme doit etre charge par une routine en Basic        *
* Etendu ou en Basic TI avec la Mini-memoire.                   *
* Le segment de programme suivant peut assurer cette tache :    *
*      10 CALL INIT                                             *
*      20 CALL LOAD("DSK1.MASTER")                              *
*      30 CALL LINK("MASTER")                                   *
* Des notes inserees dans le listing permettent de              *
* modifier ce programme pour qu'il puisse etre charge par       *
* l'option LOAD AND RUN de l'Editeur/Assembleur ou de la        *
* Mini-memoire.                                                 *
*****************************************************************
```

```
* MASTER A *
************
* La directive COPY permet d'assembler simultanement plusieurs fichiers
       COPY "DSK1.MASTERA"
       COPY "DSK1.MASTERT"
       COPY "DSK1.MASTERB"
       COPY "DSK1.MASTERC"
       COPY "DSK1.MASTERD"
       COPY "DSK1.MASTERSP1"
* version E/A ou Minimem : ajouter MASTER apres END pour obtenir l'execution
*                          automatique du programme apres chargement.
       END
************
* MASTER A *
************
       DEF  MASTER
* Version E/A ou Minimem, ajouter:
*      REF  VWTR,VSBW,VMBW,KSCAN,VSBR,VMBR *(supprimer l'asterisque)
       AORG >A000 *
VWTR   EQU  >2030  *
VSBW   EQU  >2020  *
VMBW   EQU  >2024   * Version E/A ou Minimem :
KSCAN  EQU  >201C  *  supprimer ces lignes.
VSBR   EQU  >2028  *
VMBR   EQU  >202C *
* Page pour choix du niveau. Le "-" provoque la negation du code ASCII
TEXTE  TEXT -'*************** '       * du dernier caractere (espace).
       TEXT -'* MASTER MIND * '
       TEXT -'*************** '
       TEXT -'1 -> 2 COULEURS:     32 COMBINAISONS '
       TEXT -'2 -> 3 COULEURS:    243 COMBINAISONS '
       TEXT -'3 -> 4 COULEURS:   1024 COMBINAISONS '
       TEXT -'4 -> 5 COULEURS:   3125 COMBINAISONS '
       TEXT -'5 -> 6 COULEURS:   7776 COMBINAISONS '
       TEXT -'6 -> 7 COULEURS: 16807 COMBINAISONS '
       TEXT -'7 -> 8 COULEURS: 32768 COMBINAISONS '
       TEXT -'8 -> 9 COULEURS: 59049 COMBINAISONS '
       TEXT -'9 ->10 COULEURS:100000 COMBINAISONS '
       TEXT -'ENTREZ VOTRE CHOIX... '
* Adresse dans la table image ecran pour affichage du texte.
DTEXTE DATA 13,53,93,163,243,323,403,483
       DATA 563,643,723,803,929,>FFFF
* Valeurs des registres video en mode texte.
RVDPT  DATA >0000,>01B0,>0200,>030E
       DATA >0401,>07FC,>FFFF
* Valeurs des registres video en mode Bit-Map.
RVDPBM DATA >0002,>01A1,>0206,>03FF
       DATA >0403,>0536,>0607,>0777,>FFFF
*
RVDP1T DATA >01F0          * Registre VDP 1 mode texte
RVDP1B DATA >01E1          * Registre VDP 1 mode bit-map
* Valeurs pour definitions chiffres "en toutes lettres".
ZERO   BYTE >00,>00,>00,>00,>00,>00,>00,>00
       BYTE >7E,>02,>04,>08,>10,>20,>40,>7E
       BYTE >7E,>40,>40,>78,>40,>40,>40,>7E
       BYTE >7C,>42,>42,>42,>7C,>48,>44,>42
       BYTE >7E,>42,>42,>42,>42,>42,>42,>7E
       BYTE >00,>00,>00,>00,>00,>00,>00,>00
UNE    BYTE >00,>00,>00,>00,>00,>00,>00,>00
       BYTE >04,>04,>04,>04,>04,>04,>04,>03
       BYTE >24,>26,>25,>24,>24,>24,>24,>C4
       BYTE >27,>24,>24,>A7,>64,>24,>24,>27
       BYTE >E0,>00,>00,>80,>00,>00,>00,>E0
       BYTE >00,>00,>00,>00,>00,>00,>00,>00
DEUX   BYTE >00,>00,>00,>00,>00,>00,>00,>00
       BYTE >7C,>22,>22,>22,>22,>22,>22,>7C
       BYTE >7E,>40,>40,>78,>40,>40,>40,>7E
       BYTE >42,>42,>42,>42,>42,>42,>42,>3C
       BYTE >42,>42,>24,>18,>18,>24,>42,>42
       BYTE >00,>00,>00,>00,>00,>00,>00,>00
TROIS  BYTE >03,>00,>00,>00,>00,>00,>00,>00
       BYTE >E7,>84,>84,>84,>87,>84,>84,>84
       BYTE >C7,>24,>24,>24,>C4,>84,>44,>27
       BYTE >E7,>21,>21,>21,>21,>21,>21,>E7
       BYTE >C7,>08,>08,>07,>00,>00,>08,>C7
       BYTE >80,>40,>00,>80,>40,>40,>40,>80
QUATRE BYTE >3C,>42,>42,>42,>42,>4A,>44,>3A
       BYTE >42,>42,>42,>42,>42,>42,>42,>3C
```

```
        BYTE >3C,>42,>42,>42,>7E,>42,>42,>42
        BYTE >7C,>10,>10,>10,>10,>10,>10,>10
        BYTE >F8,>84,>84,>84,>F8,>90,>88,>84
        BYTE >FC,>80,>80,>F0,>80,>80,>80,>FC
CINQ    BYTE >00,>00,>00,>00,>00,>00,>00,>00
        BYTE >3C,>42,>40,>40,>40,>40,>42,>3C
        BYTE >7C,>10,>10,>10,>10,>10,>10,>7C
        BYTE >84,>C4,>A4,>94,>8C,>84,>84,>84
        BYTE >78,>84,>84,>84,>84,>94,>88,>74
        BYTE >00,>00,>00,>00,>00,>00,>00,>00
* Adresses des blocs de valeurs.
NOMBRE DATA ZERO,UNE,DEUX,TROIS,QUATRE,CINQ
* Valeurs pour definition chiffres 7*5 pour nombre d'essais.
NES1    BYTE >00,>38,>44,>44,>44,>44,>44,>38   * 0
        BYTE >00,>10,>30,>10,>10,>10,>10,>38   * 1
NES2    BYTE >00,>38,>44,>44,>44,>44,>44,>38   * 0
        BYTE >00,>38,>44,>04,>08,>10,>20,>7C   * 2
NES3    BYTE >00,>38,>44,>44,>44,>44,>44,>38   * 0
        BYTE >00,>38,>44,>04,>18,>04,>44,>38   * 3
NES4    BYTE >00,>38,>44,>44,>44,>44,>44,>38   * 0
        BYTE >00,>08,>18,>28,>48,>7C,>08,>08   * 4
NES5    BYTE >00,>38,>44,>44,>44,>44,>44,>38   * 0
        BYTE >00,>7C,>40,>78,>04,>04,>44,>38   * 5
NES6    BYTE >00,>38,>44,>44,>44,>44,>44,>38   * 0
        BYTE >00,>18,>20,>40,>78,>44,>44,>38   * 6
NES7    BYTE >00,>38,>44,>44,>44,>44,>44,>38   * 0
        BYTE >00,>7C,>04,>08,>10,>20,>20,>20   * 7
NES8    BYTE >00,>38,>44,>44,>44,>44,>44,>38   * 0
        BYTE >00,>38,>44,>44,>38,>44,>44,>38   * 8
NES9    BYTE >00,>38,>44,>44,>44,>44,>44,>38   * 0
        BYTE >00,>38,>44,>44,>3C,>04,>08,>30   * 9
NES10   BYTE >00,>10,>30,>10,>10,>10,>10,>38   * 1
        BYTE >00,>38,>44,>44,>44,>44,>44,>38   * 0
* Adresses des blocs de valeurs.
NESSAI DATA NES1,NES2,NES3,NES4,NES5
        DATA NES6,NES7,NES8,NES9,NES10
* Valeurs pour definitions des chiffres 8*6.
C1      BYTE >08,>18,>28,>08,>08,>08,>08,>3E   * 1
C2      BYTE >3C,>42,>02,>04,>08,>10,>20,>7E   * 2
C3      BYTE >3C,>42,>02,>1C,>02,>02,>42,>3C   * 3
C4      BYTE >04,>0C,>14,>24,>44,>7E,>04,>04   * 4
C5      BYTE >7E,>40,>40,>7C,>02,>02,>42,>3C   * 5
C6      BYTE >3C,>40,>40,>7C,>42,>42,>42,>3C   * 6
C7      BYTE >7E,>02,>04,>08,>10,>20,>20,>20   * 7
C8      BYTE >3C,>42,>42,>3C,>42,>42,>42,>3C   * 8
C9      BYTE >3C,>42,>42,>42,>3E,>02,>02,>3C   * 9
C0      BYTE >3C,>42,>42,>42,>42,>42,>42,>3C   * 0
* Adresses.
CHIFFR DATA C1,C2,C3,C4,C5,C6,C7,C8,C9,C0
*
SH      BYTE >01,>03,>07,>0F,>1F,>3F,>7F,>FF   * affichage
        BYTE >1E,>21,>21,>20,>20,>20,>20,>1E   * "SOLUTION"
        BYTE >1E,>21,>21,>21,>21,>21,>21,>21   * partie
        BYTE >20,>20,>20,>20,>20,>20,>20,>20   * superieure
        BYTE >21,>21,>21,>21,>21,>21,>21,>21
        BYTE >3E,>08,>08,>08,>08,>08,>08,>08
        BYTE >7C,>10,>10,>10,>10,>10,>10,>10
        BYTE >78,>84,>84,>84,>84,>84,>84,>84
        BYTE >84,>C4,>C4,>A4,>A4,>94,>94,>8C
SB      BYTE >FF,>7F,>3F,>1F,>0F,>07,>03,>01   * affichage
        BYTE >01,>01,>01,>01,>01,>21,>21,>1E   * "SOLUTION"
        BYTE >21,>21,>21,>21,>21,>21,>21,>1E   * partie
        BYTE >20,>20,>20,>20,>20,>20,>20,>3F   * inferieure
        BYTE >21,>21,>21,>21,>21,>21,>21,>1E
        BYTE >08,>08,>08,>08,>08,>08,>08,>08
        BYTE >10,>10,>10,>10,>10,>10,>10,>7C
        BYTE >84,>84,>84,>84,>84,>84,>84,>78
        BYTE >8C,>84,>84,>84,>84,>84,>84,>84
*
ED      BYTE >00,>7C,>40,>40,>78,>40,>40,>7C   * E
        BYTE >00,>78,>44,>44,>78,>50,>48,>44   * R
        BYTE >00,>78,>44,>44,>78,>50,>48,>44   * R
        BYTE >00,>7C,>40,>40,>78,>40,>40,>7C   * E
        BYTE >00,>44,>44,>44,>44,>44,>44,>38   * U
        BYTE >00,>78,>44,>44,>78,>50,>48,>44   * R
        BYTE >00,>00,>00,>00,>00,>00,>00,>00   *
        BYTE >00,>78,>24,>24,>24,>24,>24,>78   * D
        BYTE >00,>7C,>40,>40,>78,>40,>40,>7C   * E
```

Le mode "Test" de Munch Man

Les utilisateurs du TI-99/4A qui possèdent ce module de jeu sont nombreux, et ce petit truc pourra les intéresser. Il comporte un mode test accessible par la méthode suivante : Lorsque vous sélectionnez le programme, l'image générique apparait et, quelques instants plus tard, les petits monstres commencent à s'agiter frénétiquement. Il faut agir pendant le temps où il sont immobiles, donc très rapidement. Le code *#* (SHIFT-8, SHIFT-3 et SHIFT-8) doit être tapé. L'image générique disparait et est remplacée par :

- RND (0-2)

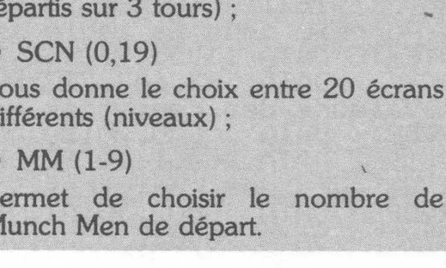
qui vous permet de choisir le tour (Munch Man comporte 60 niveaux répartis sur 3 tours) ;

- SCN (0,19)

vous donne le choix entre 20 écrans différents (niveaux) ;

- MM (1-9)

permet de choisir le nombre de Munch Men de départ.



```
*
L       BYTE >00,>40,>40,>40,>40,>40,>40,>7C   * L
        BYTE >00,>7C,>44,>44,>44,>44,>44,>7C   * O
        BYTE >00,>3C,>40,>40,>5C,>44,>44,>38   * G
        BYTE >00,>38,>10,>10,>10,>10,>10,>38   * I
        BYTE >00,>38,>44,>44,>44,>54,>48,>34   * Q
        BYTE >00,>44,>44,>44,>44,>44,>44,>38   * U
        BYTE >00,>7C,>40,>40,>78,>40,>40,>7C   * E
*
D       BYTE >00,>78,>24,>24,>24,>24,>24,>78   * D
        BYTE >00,>7C,>40,>40,>78,>40,>40,>7C   * E
        BYTE >00,>78,>44,>44,>78,>50,>48,>44   * R
        BYTE >00,>44,>64,>64,>54,>4C,>4C,>44   * N
        BYTE >00,>38,>10,>10,>10,>10,>10,>38   * I
        BYTE >00,>7C,>40,>40,>78,>40,>40,>7C   * E
        BYTE >00,>78,>44,>44,>78,>50,>48,>44   * R
*
E       BYTE >00,>7C,>40,>40,>78,>40,>40,>7C   * E
        BYTE >00,>38,>44,>40,>38,>04,>44,>38   * S
        BYTE >00,>38,>44,>40,>38,>04,>44,>38   * S
        BYTE >00,>38,>44,>44,>7C,>44,>44,>44   * A
        BYTE >00,>38,>10,>10,>10,>10,>10,>38   * I
        BYTE >00,>00,>30,>30,>00,>30,>30,>00   * :
* Definition des fleches.
FLECHE  BYTE >10,>30,>71,>FF,>FF,>71,>30,>10   * <-
        BYTE >08,>0C,>8E,>FF,>FF,>8E,>0C,>08   * ->
* Definition du cadre (curseur).
CADRE   BYTE >FE,>82,>82,>82,>82,>82,>FE,>00
* Couleurs pour tableau aide memoire.
COULR   BYTE >04,>0C,>06,>0B,>0D,>0F,>0E,>03,>09,>01
*
COULSH BYTE >06,>06,>08,>08,>09,>09,>0B,>0B   * Couleurs haut "SOLUTION"
COULSB BYTE >0A,>0A,>03,>03,>02,>02,>0C,>0C   * Couleurs bas "SOLUTION"
* Signal sonore pour erreur de logique
SONERR BYTE >05,>9F,>BF,>DF,>FF,>E3,1
        BYTE >09,>8E,>01,>A4,>02,>C5,>01,>90,>B6,>D3,6
        BYTE >03,>91,>B7,>D4,5
        BYTE >03,>92,>B8,>D5,4
        BYTE >05,>A7,>04,>93,>B0,>D6,5
        BYTE >03,>94,>B1,>D7,6
        BYTE >03,>95,>B2,>D8,7
        BYTE >05,>CA,>02,>96,>B3,>D0,6
        BYTE >03,>97,>B4,>D1,5
        BYTE >03,>98,>B5,>D2,4
        BYTE >05,>85,>03,>90,>B6,>D3,5
        BYTE >03,>91,>B7,>D4,6
        BYTE >03,>92,>B8,>D5,7
        BYTE >05,>A4,>02,>93,>B0,>D6,6
        BYTE >03,>94,>B1,>D7,5
        BYTE >03,>95,>B2,>D8,4
        BYTE >05,>C5,>01,>96,>B3,>D0,5
        BYTE >03,>97,>B4,>D1,6
        BYTE >03,>98,>B5,>D2,7
        BYTE >03,>9F,>BF,>DF,0
************
* MASTER T *
************
* Valeurs pour definition du titre
* "MASTER MIND"
TITR1   DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >FCFC,>FCFE,>FEFE,>FFFF
        DATA 0,0,0,0
        DATA >3F3F,>3F7F,>7F7F,>FFFF
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA 0,0,0,0,0,0,0,0
        DATA >0F1F,>1F1F,>3F3F,>3F7F
        DATA >FEFF,>FFFF,>FFFF,>FFFF
        DATA 0,0,>8080,>80C0
        DATA 0,0,0,0
        DATA 0,>0103,>070F,>0F1F
        DATA >1F7F,>FFFF,>FFFF,>FFFE
        DATA >FFFF,>FFFF,>FFFF,>FF07
        DATA >00C0,>F0FC,>FEFF,>FEFC
        DATA 0,0,0,0
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >FEFE,>FEFE,>FEFE,>FEFE
        DATA >0101,>0101,>0101,>0101
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >C0C0,>C0C0,>C0C0,>C0C0
        DATA >3F3F,>3F3F,>3F3F,>3F3F
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >80E0,>F8FC,>FEFE,>FFFF
        DATA 0,0,0,0
TITR2   DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >0081,>8181,>C3C3,>C3E7
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA 0,0,0,0
        DATA 0,0,>0001,>0101
        DATA >7F7F,>FFFF,>FFFF,>FEFE
        DATA >FFFF,>FFFF,>FFFF,>7F7F
        DATA >C0C0,>E0E0,>E0F0,>F0F0
        DATA 0,0,0,0
        DATA >1F1F,>1F1F,>1F1F,>1F0F
        DATA >F8F8,>FCFE,>FFFF,>FFFF
        DATA >100,0,>C0FC,>FFFF
        DATA >F0C0,0,0,>C0F0
        DATA 0,0,0,0,0,0,0,0
        DATA >3F3F,>3F3F,>3F3F,>3F3F
        DATA >F8F8,>F8F8,>F8F8,>F8F8
        DATA 0,0,0,0
        DATA >101,>101,>101,>101
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >C0C0,>C0C0,>C0C0,>FFFF
        DATA 0,0,0,>FCFC
        DATA 0,0,0,0
        DATA >3F3F,>3F3F,>3F3F,>3F3F
        DATA >F8F8,>F8F8,>F8F8,>F8F8
        DATA >703,>101,>101,>307
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >8080,>8080,>8080,>8080
```

ELSE en TI BASIC

Nous nous sommes aperçus que la plupart des programmes écrits en TI Basic comportent un grand nombre de ELSE. Il faut donc signaler que cette instruction est rarement, sinon jamais, utile dans ce langage puisque le résultat d'un test ne peut être qu'un branchement. Pour exemple, nous rencontrons souvent ce type d'écriture :

```
1000 IF X=5 THEN 1010 ELSE
     1500
1010 A=A+1
```

Ceci peut très bien, et doit, être remplacé par :

```
1000 IF X<>5 THEN 1500
1010 A=A+1
```

qui occupe moins de place en mémoire et prend moins de temps à l'exécution (le TI Basic n'a vraiment pas besoin d'être ralenti !). Ces remarques ne concernent pas, bien sûr, le Basic Etendu.

```
TITR3   DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >7F7F,>7F3F,>3F3F,>1F1F
        DATA >E7E7,>FFFF,>FFFF,>FFFF
        DATA >FBFB,>FBF3,>F3F3,>E3E3
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA 0,0,0,0
        DATA >303,>307,>707,>F0F
        DATA >FEFC,>FCFC,>F8F8,>F8FF
        DATA >7F3F,>3F3F,>1F1F,>1FFF
        DATA >F8F8,>F8FC,>FCFC,>FEFE
        DATA 0,0,0,0
        DATA >F07,>301,0,0
        DATA >FFFF,>FFFF,>FF3F,>700
        DATA >FFFF,>FFFF,>FFFF,>FF7F
        DATA >F8FC,>FEFF,>FFFF,>FFFF
        DATA 0,0,>80,>8080
        DATA 0,0,0,0
        DATA >3F3F,>3F3F,>3F3F,>3F3F
        DATA >F8F8,>F8F8,>F8F8,>F8F8
        DATA 0,0,0,0
        DATA >101,>101,>101,>101
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >FFFF,>FFFF,>FFFF,>C0C0
        DATA >FCFC,>FCFC,>FCFC,0
        DATA 0,0,0,0
        DATA >3F3F,>3F3F,>3F3F,>3F3F
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >FFFF,>FEFC,>F8F0,>C0E0
        DATA 0,0,0,0
TITR4   DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >1F0F,>F0F,>707,>703
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >E3C3,>C3C3,>8383,>8303
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA 0,0,0,0
        DATA >F1F,>1F1F,>3F3F,>3F7F
        DATA >FFFF,>FFFF,>FFFF,>FFC0
        DATA >FFFF,>FFFF,>FFFF,>FF03
        DATA >FEFF,>FFFF,>FFFF,>FFFF
        DATA 0,0,>8080,>80C0
        DATA 0,>030F,>3F7F,>3F1F
        DATA >80,>C0E0,>F0FC,>FFFF
        DATA >703,>101,>30F,>FFFF
        DATA >FFFF,>FFFF,>FFFF,>FEFE
        DATA >8080,>8080,0,0,0,0,0,0
        DATA >3F3F,>3F3F,>3F3F,>3F3F
        DATA >F8F8,>F8F8,>F8F8,>F8F8
        DATA 0,0,0,0
        DATA >101,>101,>101,>101
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >C0C0,>C0C0,>C0FF,>FFFF
        DATA 0,0,>FF,>FFFF
        DATA 0,0,>C0,>C0C0
        DATA >3F3F,>3F3F,>3F3F,>3F3F
        DATA >F9F8,>F8F8,>F8F8,>F8F8
        DATA >FFFF,>FF7F,>7F3F,>3F1F
        DATA >E0F0,>F0F8,>F8FC,>FCFE
        DATA 0,0,0,0
TITR5   DATA >FFFF,>FFFF,>FF00,0
        DATA >303,>101,>100,0
        DATA >FFFF,>FEFE,>FE00,0
        DATA >303,>303,>300,0
        DATA >FFFF,>FFFF,>FF00,0
        DATA 0,0,0,0
        DATA >7F7F,>FFFF,>FF00,0
        DATA >C0C0,>8080,>8000,0
        DATA >303,>101,>100,0
        DATA >FFFF,>FFFF,>FF00,0
        DATA >C0C0,>E0E0,>E000,0
        DATA >F07,>300,0,0
        DATA >FFFF,>FFFF,>1F00,0
        DATA >FFFF,>FFFF,>FE00,0
        DATA >FCF8,>F0C0,0,0
        DATA 0,0,0,0,0,0,0,0
        DATA >3F3F,>3F3F,>3F00,0
        DATA >F8F8,>F8F8,>F800,0
        DATA 0,0,0,0,>101,>101,>100,0
        DATA >FFFF,>FFFF,>FF00,0
        DATA >FFFF,>FFFF,>FF00,0
        DATA >FFFF,>FFFF,>FF00,0
        DATA >C0C0,>C0C0,>C000,0
        DATA >3F3F,>3F3F,>3F00,0
        DATA >F8F8,>F8F8,>F800,0
        DATA >1F0F,>F07,>700,0
        DATA >FEFF,>FFFF,>FF00,0
        DATA 0,>80,>8000,0
TITR6   DATA >101,>101,>101,>101
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >F8F8,>F8FC,>FCFC,>FEFE
        DATA 0,0,0,>101
        DATA >7F7F,>7FFF,>FFFF,>FFFF
        DATA >FEFE,>FEFE,>FEFE,>FEFE
        DATA >101,>101,>101,>101
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >C0C0,>C0C0,>C0C0,>C0C0
        DATA >3F3F,>3F3F,>3F3F,>3F3F
        DATA >F0F8,>F8FC,>FEFE,>FFFF
        DATA 0,0,0,>80
        DATA >3F3F,>3F3F,>3F3F,>3F3F
        DATA >C0C0,>C0C0,>C0C0,>C0C0
        DATA >3F3F,>3F3F,>3F3F,>3F3F
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >F8FF,>FFFF,>FFFF,>FFFF
        DATA 0,>C0E0,>F0F8,>FCFC
        DATA 0,0,0,0
TITR7   DATA >101,>101,>101,>101
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >FEFF,>FFFF,>FFFF,>FFFF
        DATA >103,>303,>8787,>87CF
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >FEFE,>FEFE,>FEFE,>FEFE
        DATA >101,>101,>101,>101
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >C0C0,>C0C0,>C0C0,>C0C0
        DATA >3F3F,>3F3F,>3F3F,>3F3F
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >80C0,>E0E0,>F0F8,>F8FC
        DATA >3F3F,>3F3F,>3F3F,>3F3F
        DATA >C0C0,>C0C0,>C0C0,>C0C0
        DATA >3F3F,>3F3F,>3F3F,>3F3F
        DATA >F0F0,>F0F0,>F0F0,>F0F0
        DATA >1F07,>301,>101,>101
        DATA >FEFE,>FFFF,>FFFF,>FFFF
        DATA 0,0,>80,>8080
TITR8   DATA >101,>101,>101,>101
        DATA >FEFE,>FEFE,>FEFE,>FEFE
        DATA >FFFF,>FF7F,>7F7F,>3F3F
        DATA >CFCF,>FFFF,>FFFF,>FFFF
        DATA >F7F7,>F7E7,>E7E7,>C7C7
        DATA >FEFE,>FEFE,>FEFE,>FEFE
        DATA >101,>101,>101,>101
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >C0C0,>C0C0,>C0C0,>C0C0
        DATA >3F3F,>3F3F,>3F3F,>3F3F
        DATA >FFFF,>DFDF,>CFC7,>C7C3
        DATA >FEFE,>FFFF,>FFFF,>FFFF
        DATA >3F3F,>3FBF,>BFFF,>FFFF
        DATA >C0C0,>C0C0,>C0C0,>C0C0
        DATA >3F3F,>3F3F,>3F3F,>3F3F
        DATA >F0F0,>F0F0,>F0F0,>F0F0
        DATA >101,>101,>101,>101
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >8080,>8080,>8080,>8080
TITR9   DATA >101,>101,>101,>101
        DATA >FEFE,>FEFE,>FEFE,>FEFE
        DATA >3F1F,>1F1F,>F0F,>F07
        DATA >FFFF,>FFFF,>FFFF,>FFFE
        DATA >C787,>8787,>707,>707
        DATA >FEFE,>FEFE,>FEFE,>FEFE
        DATA >101,>101,>101,>101
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >C0C0,>C0C0,>C0C0,>C0C0
        DATA >3F3F,>3F3F,>3F3F,>3F3F
        DATA >C1C1,>C0C0,>C0C0,>C0C0
        DATA >FFFF,>FF7F,>7F3F,>1F1F
```

```
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >C0C0,>C0C0,>C0C0,>C0C0
        DATA >3F3F,>3F3F,>3F3F,>3F3F
        DATA >F0F0,>F0F0,>F0FF,>FFFF
        DATA >101,>307,>1FFF,>FFFF
        DATA >FFFF,>FFFE,>FEFC,>FCF8
        DATA 0,0,0,0
TITR10  DATA >101,>101,>100,0
        DATA >FEFE,>FEFE,>FE00,0
        DATA >707,>303,>300,0
        DATA >FEFE,>FCFC,>FC00,0
        DATA >707,>707,>700,0
        DATA >FEFE,>FEFE,>FE00,0
        DATA >101,>101,>100,0
        DATA >FFFF,>FFFF,>FF00,0
        DATA >C0C0,>C0C0,>C000,0
        DATA >3F3F,>3F3F,>3F00,0
        DATA >C0C0,>C0C0,>C000,0
        DATA >F07,>703,>100,0
        DATA >FFFF,>FFFF,>FF00,0
        DATA >C0C0,>C0C0,>C000,0
        DATA >3F3F,>3F3F,>3F00,0
        DATA >FFFF,>FFFF,>FF00,0
        DATA >FFFF,>FFFF,>F800,0
        DATA >F0E0,>C000,0,0
        DATA 0,0,0,0
TITR11  DATA >78,>4444,>7840,>4040
        DATA >7C,>4444,>4444,>447C
        DATA >44,>4444,>4444,>4438
        DATA >78,>4444,>7850,>4844
        DATA 0,0,0,0
        DATA >38,>4440,>4040,>4438
        DATA >7C,>4444,>4444,>447C
        DATA >44,>6C54,>5444,>4444
        DATA >44,>6C54,>5444,>4444
        DATA >7C,>4040,>7840,>407C
        DATA >44,>6464,>544C,>4C44
        DATA >38,>4440,>4040,>4438
        DATA >7C,>4040,>7840,>407C
        DATA >78,>4444,>7850,>4844
        DATA 0,0,>38,>1020
        DATA 0,0,0,0
        DATA >38,>4444,>7C44,>4444
        DATA >78,>4444,>7840,>4040
        DATA >78,>4444,>7840,>4040
        DATA >44,>4444,>4444,>4438
        DATA >44,>4428,>1010,>1010
        DATA >7C,>4040,>7840,>407C
        DATA >7C,>0408,>1020,>407C
TITR12  DATA >38,>4440,>3804,>4438
        DATA >44,>4444,>4444,>4438
        DATA >78,>4444,>7850,>4844
        DATA 0,0,0,0
        DATA >44,>6464,>544C,>4C44
        DATA 8,>810,0,0
        DATA >38,>1010,>1010,>1038
        DATA >44,>6C54,>5444,>4444
        DATA >78,>4444,>7840,>4040
        DATA >7C,>4444,>4444,>447C
        DATA >78,>4444,>7850,>4844
        DATA >7C,>1010,>1010,>1010
        DATA >7C,>4040,>7840,>407C
        DATA 0,0,0,0
        DATA >38,>4444,>4454,>4834
        DATA >44,>4444,>4444,>4438
        DATA >7C,>4040,>7840,>407C
        DATA >40,>4040,>4040,>407C
        DATA >40,>4040,>4040,>407C
        DATA >7C,>4040,>7840,>407C
        DATA 0,0,0,0
        DATA >7C,>1010,>1010,>1010
        DATA >7C,>4444,>4444,>447C
        DATA >44,>4444,>4444,>4438
        DATA >38,>4440,>4040,>4438
        DATA >44,>4444,>7C44,>4444
        DATA >7C,>4040,>7840,>407C
* Adresses des blocs pour titre.
ATTITR  DATA TITR1,TITR2,TITR3,TITR4
        DATA TITR5
        DATA TITR6,TITR7,TITR8,TITR9
        DATA TITR10,TITR11,TITR12
* Adresses dans table image ecran.
AMTITR  DATA >210,>310,>410,>510
        DATA >610,>B38,>C38
        DATA >D38,>E38,>F38,>1528
        DATA >1718
* Nombre d'octets par blocs.
NOCTTI  DATA 240,240,240,240,240
        DATA 152,152
        DATA 152,152,152,184,216,0
* Couleurs pour titre (scroll).
COLTIT  DATA >6060,>8080,>9090,>B0B0
        DATA >A0A0,>3030,>2020,>C0C0
        DATA >6060,>8080,>9090,>B0B0
        DATA >A0A0,>3030,>2020,>C0C0

************
* MASTER B *
************
        EVEN
R       BYTE >00,>00,>00,>00,>00,>00,>00,>00
        BYTE >00,>00,>00,>00,>00,>00,>00,>00
        BYTE >00,>78,>44,>44,>78,>50,>48,>44    * R
        BYTE >00,>7C,>40,>40,>78,>40,>40,>7C    * E
        BYTE >00,>78,>24,>24,>24,>24,>24,>78    * D
        BYTE >00,>7C,>44,>44,>44,>44,>44,>7C    * O
        BYTE >00,>00,>00,>00,>00,>00,>00,>00
*
OB      BYTE >00,>7C,>44,>44,>44,>44,>44,>7C    * O
        BYTE >00,>44,>44,>44,>44,>44,>44,>38    * U
        BYTE >00,>00,>00,>00,>00,>00,>00,>00
        BYTE >00,>00,>00,>00,>00,>00,>00,>00
        BYTE >00,>78,>24,>24,>38,>24,>24,>78    * B
        BYTE >00,>38,>44,>44,>7C,>44,>44,>44    * A
        BYTE >00,>38,>44,>40,>40,>40,>44,>38    * C
        BYTE >00,>44,>48,>50,>60,>50,>48,>44    * K
*
TBACK   BYTE 15                 * touche BACK
TREDO   BYTE 6                  *    "    REDO
TQUIT   BYTE 5                  *    "    QUIT
TAID    BYTE 1                  *    "    AID
TENTER  BYTE 13                 *    "    ENTER
TG      BYTE 83                 *    "    <-
TD      BYTE 68                 *    "    ->
TE      BYTE 69                 *    "    E
TO      BYTE 48                 *    "    ZERO
```

Impression plus rapide des fichiers-source avec l'Editeur/Assembleur.

L'option PRINT de l'éditeur envoie le texte vers l'imprimante en format fixe de 80 octets. Ceci provoque le déplacement systématique du chariot jusqu'à la 80ème colonne, même si la ligne ne contient qu'un caractère. L'option SAVE, utilisée normalement pour la sauvegarde sur disquette, permet la transmission des données en format variable.

Il est nécessaire de charger le fichier source en mémoire pour utiliser l'option SAVE, mais le temps gagné à l'impression est considérable, surtout si l'on utilise une imprimante lente. A la question "FILE NAME ?", il suffit de répondre "PIO.LF" (pour une imprimante parallèle) au lieu de "DSK1.nom de fichier".

```
*
DCADRE BYTE >04,>05,>00,>0F,>D0              * Table de description des lutins
*
C1CADR BYTE >05
CADREY BYTE >04             * Position cadre Y (Ligne)
CADREX BYTE >05             * Position cadre X (Colonne)
AJCADR BYTE >10             * Valeur deplacement cadre
H01    BYTE >01             * Drapeau pour son
       EVEN
*
TASCII BSS  472             * Tampon pour stockage definitions caracteres standards
NBIENP BSS  20              * Nombre de couleurs bien placees
NMALP  BSS  20              * Nombre de couleurs mal placee
STES1  BSS  10              * Stockage essai 1
STES2  BSS  10              *    "      "   2
STES3  BSS  10              *    "      "   3
STES4  BSS  10              *    "      "   4
STES5  BSS  10              *    "      "   5
STES6  BSS  10              *    "      "   6
STES7  BSS  10              *    "      "   7
STES8  BSS  10              *    "      "   8
STES9  BSS  10              *    "      "   9
STES10 BSS  10              *    "      "   10
STINT  BSS  10              * Stockage intermediaire
STINTS BSS  10              * Stockage intermediaire solution
STSOL  BSS  10              * Stockage solution
REGPP  BSS  >20             * Memoire de travail du programme principal
REGSP  BSS  >20             * Memoire de travail sous programmes
*
NCOUL  DATA 0               * Stockage choix du nombre de couleurs
ADRRET DATA 0               * Adresse de retour GPL
DRAPPO DATA 0               * Drapeau pour test 5 positions occupees
DRAPT1 DATA 0               * Drapeau pour test si premiere touche pressee
DRAPER DATA 0               * Drapeau pour erreur de logique
AECESR DATA 0               * Adresse relative pour ecriture dans table couleurs
AECRBP DATA >0158           * Adresse ecriture resultat  couleurs bien placees
AECRMP DATA >01A8           *    "        "       "      couleurs mal placees
AJ200  DATA >0200           * Constante pour deplacement dans memoire video
C5COUL DATA 10              * Pour test si 5 couleurs trouvees
*
H4000  DATA >4000
H0007  DATA >0007           * Constantes
HF800  DATA >F800           * pour tests divers
*
ASTES  DATA STES1,STES2,STES3,STES4,STES5    * Adresses des tampons de
       DATA STES6,STES7,STES8,STES9,STES10   * stockage des essais
*
AECRES DATA >2108,>2308,>2508,>2708,>2908    * Adresses premieres
       DATA >2B08,>2D08,>2F08,>3108,>3308    * Positions couleurs
*
AECRFL DATA >2190,>2390,>2590,>2790,>2990    * Adresses fleches dans
       DATA >2B90,>2D90,>2F90,>3190          * tables couleurs
*
ETAT   EQU  >837C           * Registre d'etat GPL
RTCLAV EQU  >8375           * Adresse retour code ASCII (clavier)
NOMBAL EQU  >83C0           * Nombre aleatoire
REG5   EQU  REGPP+10        * Registre 5 du programme principal
REG6   EQU  REGPP+12        *    "     6      "         "
REG7   EQU  REGPP+14        *    "     7      "         "
REG8   EQU  REGPP+16        *    "     8      "         "
REG9   EQU  REGPP+18        *    "     9      "         "
* sauvegarde adresse retour
* charge registre pointeur
* memoire de travail
* sauvegarde definition des
* caracteres dans tampon
MASTER MOV  R11,@ADRRET
       LWPI REGPP
* Version E/A ou minimem :
* remplacer R0,>0400 par
* R0,>0900
       LI   R0,>0400
       LI   R1,TASCII
       LI   R2,472
       BLWP @VMBR
* passage en mode Bit-Map
* affichage du titre
BACK2  MOVB @RVDPBM+3,@>83D4
       LI   R1,RVDPBM
```

Des minuscules avec CTRL

Les minuscules du TI-99 sont assez décevantes : leur air de famille avec leurs grandes soeurs est par trop flagrant. Pour stocker du texte, on peut se "fabriquer" son propre jeu de minuscules au moyen de l'instruction CALL CHAR. Sachant que la pression de CTRL-A, par exemple, provoque l'affichage du caractère de code 129, CTRL-B celui du caractère 130 et ainsi de suite... on peut afficher ces minuscules le plus simplement du monde. Pour obtenir un "a" il suffira d'appuyer simultanément sur CTRL et A, à condition évidemment d'avoir préalablement défini le caractère 129 avec l'instruction CALL CHAR. Remarque : lors d'une utilisation ultérieure, pour qu'un texte stocké sous cette forme soit lisible il faudra bien sûr redéfinir les caractères 129 à 154.

```
IRBM2  MOV   *R1+,R0
       JLT   INITL2
       BLWP  @VWTR
       JMP   IRBM2
INITL2 LI    R0,>1B00
       LI    R1,>D000
       BLWP  @VSBW
       LI    R0,>0711
       BLWP  @VWTR
       CLR   R2
       LI    R0,>1800
ITIE12 CLR   R1
ITIE22 SWPB  R1
       BLWP  @VSBW
       SWPB  R1
       INC   R1
       INC   R0
       CI    R1,>100
       JLT   ITIE22
       INC   R2
       CI    R2,3
       JLT   ITIE12
       CLR   R0
       CLR   R1
       LI    R2,>1800
       BL    @APOV
       LI    R0,>2000
       CLR   R1
       LI    R2,>1800
       BL    @APOV
       CLR   R3
DTITRE MOV   @AMTITR(R3),R0
       MOV   @ATTITR(R3),R1
       MOV   @NOCTTI(R3),R2
       JEQ   CTITR
       BLWP  @VMBW
       INCT  R3
       JMP   DTITRE
CTITR  LI    R0,>3500
       LI    R1,>7000
       LI    R2,768
       BL    @APOV
* affichage permis
       MOVB  @RVDP1B+1,@>83D4
       MOV   @RVDP1B,R0
       BLWP  @VWTR
* defilement des couleurs
* teste clavier pour
* controle touche pressee
       LI    R2,8
CTITR7 SETO  R3
CTITR6 INC   R3
       LI    R4,-2
       BLWP  @KSCAN
       MOVB  @ETAT,R0
       JEQ   CTITR5
       LI    R0,>900
       LI    R1,TASCII
       LI    R2,472
       BLWP  @VMBW
       B     @BACK
CTITR5 LI    R1,COLTIT
       A     R3,R1
       INCT  R4
       MOV   @AMTITR(R4),R0
       AI    R0,>2000
       MOV   R0,R5
       A     @NOCTTI(R4),R5
CTITR1 BLWP  @VMBW
       AI    R0,8
       C     R0,R5
       JGT   CTITR2
       JMP   CTITR1
CTITR2 LI    R1,COLTIT+8
       A     R3,R1
       INCT  R4
       MOV   @AMTITR(R4),R0
       AI    R0,>2000
       MOV   R0,R5
       A     @NOCTTI(R4),R5
CTITR4 BLWP  @VMBW
       AI    R0,8
       C     R0,R5
       JGT   CTITR3
       JMP   CTITR4
CTITR3 CI    R4,18
       JNE   CTITR5
       CI    R3,15
       JNE   CTITR6
       JMP   CTITR7
* passage en mode texte
* affichage choix niveau
BACK   MOVB  @RVDPT+3,@>83D4
       LI    R1,RVDPT
IRT    MOV   *R1+,R0
       JLT   CHOIX
       BLWP  @VWTR
       JMP   IRT
*
CHOIX  LI    R1,>2000
       LI    R2,960
       CLR   R0
       BL    @APOV
       LI    R3,DTEXTE
       LI    R2,TEXTE
CHOIX2 MOV   *R3+,R0
       JLT   CHOIXC
CHOIX3 MOVB  *R2+,R1
       JLT   CHOIX2
       BLWP  @VSBW
       INC   R0
       JMP   CHOIX3
* affichage permis
CHOIXC MOVB  @RVDP1T+1,@>83D4
       MOV   @RVDP1T,R0
       BLWP  @VWTR
       SETO  R0
DELAIX NOP
       DEC   R0
       JNE   DELAIX
* scrutation du clavier
* pour determiner
* choix niveau de jeu
* si touche QUIT enfoncee :
* retour a la mire
CLAV1  BLWP  @KSCAN
       CB    @RTCLAV,@TQUIT
       JNE   CLAV1A
       B     @EXIT
CLAV1A MOVB  @RTCLAV,R0
       SRL   R0,8
       AI    R0,-47
       CI    R0,2
       JLT   CLAV1
       CI    R0,10
       JGT   CLAV1
       MOV   R0,@NCOUL
**************************
* initialisation du jeu *
**************************
* affichage annule
REDO   MOVB  @RVDPBM+3,@>83D4
       LI    R1,RVDPBM
* passage en mode Bit-Map
IRBM   MOV   *R1+,R0
       JLT   INITL
       BLWP  @VWTR
       JMP   IRBM
INITL  LI    R0,>0158
       MOV   R0,@AECRBP
* initialisations pour cadre
* table formes lutins: >3800
* table definitions lutins: >1B00
       LI    R0,>3800
       LI    R1,CADRE
       LI    R2,8
```

```
        BLWP @VMBW
        LI   R0,>1B00
        LI   R1,DCADRE
        LI   R2,5
        BLWP @VMBW
* initialisation table image ecran
* adresse de base = >1800
        LI   R0,>1800
        CLR  R2
ITIE1   CLR  R1
ITIE2   SWPB R1
        BLWP @VSBW
        SWPB R1
        INC  R1
        INC  R0
        CI   R1,>100
        JLT  ITIE2
        INC  R2
        CI   R2,3
        JLT  ITIE1
* mise a zero table des formes
* adresse de base = >0000
        CLR  R0
        CLR  R1
        LI   R2,>1800
        BL   @APOV
* table des couleurs = transparent
* adresse de base = >2000
        LI   R0,>2000
        CLR  R1
        LI   R2,>1800
        BL   @APOV
* formation des "cases" destinees
* a recevoir les couleurs
        LI   R5,>1000
        LI   R0,10
        LI   R7,7
        MOV  R7,R9
CAR4    LI   R6,7
        LI   R8,16
CAR2    BLWP @LIGNE
        CI   R6,71
        JEQ  CAR1
        AI   R6,16
        AI   R8,16
        JMP  CAR2
CAR1    COC  @H0007,R7
        JNE  CAR3
        AI   R7,9
        MOV  R7,R9
        JMP  CAR4
CAR3    AI   R7,7
        MOV  R7,R9
        DEC  R0
        JNE  CAR4
        LI   R0,5
        LI   R6,7
        MOV  R6,R8
        BL   @SCAR
        LI   R0,1
        LI   R6,223
        MOV  R6,R8
        BL   @SCAR
        LI   R0,10
        LI   R6,223
        LI   R7,7
        LI   R8,232
        MOV  R7,R9
CAR5    BLWP @LIGNE
        COC  @H0007,R7
        JNE  CAR6
        AI   R7,9
        MOV  R7,R9
        JMP  CAR5
CAR6    AI   R7,7
        MOV  R7,R9
        DEC  R0
        JNE  CAR5
        LI   R7,167
        MOV  R7,R9
CAR7    LI   R6,7
        LI   R8,16
CAR8    BLWP @LIGNE
        CI   R6,71
        JEQ  CAR9
        AI   R6,16
        AI   R8,16
        JMP  CAR8
CAR9    CI   R7,184
        JEQ  CAR10
        LI   R7,184
        MOV  R7,R9
        JMP  CAR7
CAR10   LI   R0,5
        LI   R6,7
        MOV  R6,R8
        LI   R7,168
        LI   R9,183
CAR11   BLWP @LIGNE
        COC  @H0007,R6
        JNE  CAR12
        AI   R6,9
        MOV  R6,R8
        JMP  CAR11
CAR12   AI   R6,7
        MOV  R6,R8
        DEC  R0
        JNE  CAR11
************
* MASTER C *
************
* Trace les lignes
        LI   R6,3
        MOV  R6,R8
        LI   R7,5
        LI   R9,186
        BLWP @LIGNE
        INC  R6
        INC  R8
        BLWP @LIGNE
        LI   R6,83
        MOV  R6,R8
        BLWP @LIGNE
        INC  R6
        INC  R8
        BLWP @LIGNE
        LI   R6,251
        MOV  R6,R8
        BLWP @LIGNE
        INC  R6
        INC  R8
        BLWP @LIGNE
        LI   R6,219
        MOV  R6,R8
        LI   R9,162
        BLWP @LIGNE
        INC  R6
        INC  R8
        BLWP @LIGNE
*
        LI   R5,>3000
        LI   R6,139
        MOV  R6,R8
        BLWP @LIGNE
        INC  R6
        INC  R8
        BLWP @LIGNE
        LI   R6,163
        MOV  R6,R8
        LI   R9,186
        BLWP @LIGNE
        INC  R6
        INC  R8
        BLWP @LIGNE
*
        LI   R5,>4000
```

```
       LI   R6,3
       MOV  R6,R7
       MOV  R6,R9
       LI   R8,252
       BLWP @LIGNE
       INC  R7
       INC  R9
       BLWP @LIGNE
       LI   R7,187
       MOV  R7,R9
       BLWP @LIGNE
       INC  R7
       INC  R9
       BLWP @LIGNE
       INCT R6
       LI   R7,163
       MOV  R7,R9
       DECT R8
       BLWP @LIGNE
       INC  R7
       INC  R9
       BLWP @LIGNE
*
       LI   R5,>5000
       LI   R7,19
       MOV  R7,R9
       LI   R8,218
       LI   R0,9
LVERTE BLWP @LIGNE
       INC  R7
       INC  R9
       BLWP @LIGNE
       AI   R7,15
       MOV  R7,R9
       DEC  R0
       JNE  LVERTE
* initialisation couleur
* "SOLUTION"
* partie superieure
       LI   R3,5
       LI   R0,>3508
       LI   R1,COULSH
       LI   R2,8
ICSH   BLWP @VMBW
       AI   R0,>10
       DEC  R3
       JNE  ICSH
* initialisation couleur
* "SOLUTION"
* partie inferieure
       LI   R3,5
       LI   R0,>3608
       LI   R1,COULSB
ICSB   BLWP @VMBW
       AI   R0,>10
       DEC  R3
       JNE  ICSB
* initialisation couleur
* nombre bien placees
       LI   R3,10
       LI   R0,>2158
       LI   R1,>4000
ICCBP  LI   R2,48
       BL   @APOV
       AI   R0,>200
       DEC  R3
       JNE  ICCBP
* initialisation couleur
* nombre mal placees
       LI   R3,10
       LI   R0,>21A8
       LI   R1,>C000
ICCMP  LI   R2,48
       BL   @APOV
       AI   R0,>200
       DEC  R3
       JNE  ICCMP
* initialisation fleches
* pour erreur de logique
       LI   R3,10
       LI   R0,>0190
       LI   R1,FLECHE
       LI   R2,16
IFLECH BLWP @VMBW
       AI   R0,>200
       DEC  R3
       JNE  IFLECH
* initialisation couleurs
* du tableau aide memoire
       LI   R0,>21E0
ICREF  MOVB @COULR(R3),R1
       LI   R2,8
       BL   @APOV
       AI   R0,>200
       INC  R3
       C    R3,@NCOUL
       JLT  ICREF
* affichage des chiffres
* 8*6 du tableau
* aide memoire
       CLR  R4
       CLR  R3
       LI   R0,>01F0
       LI   R2,8
IDREF  MOV  @CHIFFR(R3),R1
       BLWP @VMBW
       AI   R0,>200
       INCT R3
       INC  R4
       C    R4,@NCOUL
       JLT  IDREF
*
       LI   R3,10
       LI   R0,>21F0
       LI   R1,>1000
IDCREF LI   R2,8
       BL   @APOV
       AI   R0,>200
       DEC  R3
       JNE  IDCREF
* initialisation affichage
* "ERREUR DE LOGIQUE"
       LI   R0,>1558
       LI   R1,ED
       LI   R2,72
       BLWP @VMBW
*
       LI   R0,>1660
       LI   R1,L
       LI   R2,56
       BLWP @VMBW
* initialisation affichage
* "DERNIER ESSAI"
       LI   R0,>15B0
       LI   R1,D
       BLWP @VMBW
*
       LI   R0,>16B0
       LI   R1,E
       LI   R2,48
       BLWP @VMBW
*
       LI   R0,>36B0
       LI   R1,>4000
       LI   R2,48
       BL   @APOV
*
       LI   R0,>36E0
       LI   R1,>D000
       LI   R2,16
       BL   @APOV
* Chargement table des
* sons dans mem video
       LI   R0,>3C00
       LI   R1,SONERR
       LI   R2,118
```

```
       BLWP @VMBW
* tirage aleatoire des
* couleurs a decouvrir
       LI   R0,STSOL
CODE   BL   @SPALEA
       MOV  @NOMBAL,R1
       SRL  R1,12
       C    R1,@NCOUL
       JHE  CODE
       MOV  R1,*R0+
       CI   R0,STSOL+10
       JLT  CODE
* mise a zero des tampons
* de stockage des essais
       LI   R2,40
       LI   R1,NBIENP
       CLR  R0
MISEZ  MOV  R0,*R1+
       DECT R2
       JNE  MISEZ
* initialisation pointeurs
       SETO R3
       LI   R4,-2
* affichage permis
       MOVB @RVDP1B+1,@>83D4
       MOV  @RVDP1B,R0
       BLWP @VWTR
***********
*MASTER D *
***********
* Initialisations pour
* premier ou nouvel essai
NOUVES INC  R3
       INCT R4
       CLR  R5
       LI   R0,>16E0
       MOV  @NESSAI(R4),R1
       LI   R2,16
       BLWP @VMBW
       CLR  @DRAPT1
       CLR  @DRAPPO
       MOV  @AECRES(R4),R12
       MOV  @ASTES(R4),R10
       MOV  @DRAPER,@DRAPER
* si erreur de logique
* declenchement signal
* sonore, sinon aller vers
* scrutation du clavier
       JEQ  CLAV2
       LI   R0,>3558
       LI   R1,>6000
       LI   R2,72
       BL   @APOV
       LI   R0,>3660
       LI   R2,56
       BL   @APOV
       LI   R0,>3C00
       MOV  R0,@>83CC
       SOCB @H01,@>83FD
       MOVB @H01,@>83CE
* scrutation du clavier pour
* saisie couleurs et
* deplacements cadre
* les interruptions permettent
* le signal sonore pendant la
* boucle
CLAV2  LIMI 2
       LIMI 0
       BLWP @KSCAN
       MOVB @ETAT,R0
       JEQ  CLAV2
       MOV  @DRAPT1,@DRAPT1
       JNE  CLAV2A
       BL   @EFFLE
       SETO @DRAPT1
* test si deplacement a gauche
CLAV2A CB   @RTCLAV,@TG
       JNE  CLAV2B
       B    @GAUCH
* test si deplacement a droite
CLAV2B CB   @RTCLAV,@TD
       JNE  CLAV2H
       B    @DROIT
* test si effacement
CLAV2H CB   @RTCLAV,@TE
       JNE  CLAV2C
       B    @EFFAC
* test si <ENTER>
CLAV2C CB   @RTCLAV,@TENTER
       JNE  CLAV2D
       B    @ENTRE
* test si abandon (AID)
CLAV2D CB   @RTCLAV,@TAID
       JNE  CLAV2E
       CLR  R15
       B    @ABAND
* test si couleur = noir
CLAV2E CB   @RTCLAV,@T0
       JNE  CLAV2F
       LI   R9,9
       JMP  CLAV2G
* tests si entree correcte
CLAV2F MOVB @RTCLAV,R9
       SRL  R9,8
       AI   R9,-49
       JLT  CLAV2
* affiche la couleur et
* deplace le cadre vers la
* droite si possible
CLAV2G C    R9,@NCOUL
       JHE  CLAV2
       MOV  R9,*R10
       MOV  R12,R0
       MOVB @COULR(R9),R1
       LI   R2,8
       BL   @APOV
       MOV  R5,R0
       LI   R7,>8000
       SRC  R7,0
       SOC  R7,@DRAPPO
       B    @DROIT
* test si les 5 positions
* sont occupees
ENTRE  MOV  @DRAPPO,R7
       COC  @HF800,R7
       JEQ  ENTREA
       B    @CLAV2
* determine le nombre de
* couleurs bien placees
ENTREA MOV  @ASTES(R4),R10
       CLR  R7
ENTREG MOV  *R10+,@STINT(R7)
       MOV  @STSOL(R7),@STINTS(R7)
       INCT R7
       CI   R7,10
       JLT  ENTREG
       CLR  R7
ENTREC C    @STINTS(R7),@STINT(R7)
       JNE  ENTREB
       INCT @NBIENP(R4)
       MOV  @HF800,@STINTS(R7)
       SETO @STINT(R7)
ENTREB INCT R7
       CI   R7,10
       JLT  ENTREC
* determine le nombre de
* couleurs mal placees
       CLR  R7
ENTREF CLR  R8
ENTREE C    @STINTS(R8),@STINT(R7)
       JNE  ENTRED
       INCT @NMALP(R4)
       MOV  @HF800,@STINTS(R8)
       SETO @STINT(R7)
ENTRED INCT R8
       CI   R8,10
```

```
       JLT   ENTREE
       INCT  R7
       CI    R7,10
       JLT   ENTREF
* affiche le nombre de couleurs
* bien et mal placees
       MOV   @NBIENP(R4),R7
       MOV   @NOMBRE(R7),R1
       MOV   @AECRBP,R0
       LI    R2,48
       BLWP  @VMBW
       MOV   @NMALP(R4),R7
       MOV   @NOMBRE(R7),R1
       MOV   @AECRMP,R0
       BLWP  @VMBW
* test si 5 couleurs trouvees
       C     @NBIENP(R4),@C5COUL
       JNE   ENTREH
       SETO  R15
       B     @ABAND
ENTREH MOV   R4,R4
       JNE   TESTEX
       B     @SORNE
* routine pour determiner
* si le joueur a fait une
* erreur de logique
TESTEX LI    R0,>2190
       LI    R1,>F000
       CLR   @DRAPER
       MOV   @ASTES(R4),R6
       CLR   R7
TESTE  CLR   R14
       CLR   R15
       MOV   @ASTES(R7),R10
       MOV   @ASTES(R4),R6
       CLR   R13
TESTEA MOV   *R10+,@STINT(R13)
       MOV   *R6+,@STINTS(R13)
       INCT  R13
       CI    R13,10
       JLT   TESTEA
       CLR   R8
TESTEB C     @STINTS(R8),@STINT(R8)
       JNE   TESTEC
       INCT  R14
       MOV   @HF800,@STINTS(R8)
       SETO  @STINT(R8)
TESTEC INCT  R8
       CI    R8,10
       JLT   TESTEB
       CLR   R8
TESTEF CLR   R9
TESTEE C     @STINTS(R9),@STINT(R8)
       JNE   TESTED
       INCT  R15
       MOV   @HF800,@STINTS(R9)
       SETO  @STINT(R8)
TESTED INCT  R9
       CI    R9,10
       JLT   TESTEE
       INCT  R8
       CI    R8,10
       JLT   TESTEF
       C     R14,@NBIENP(R7)
       JEQ   TESTEG
       MOV   @AECRFL(R7),R0
       LI    R2,8
       BL    @APOV
       SETO  @DRAPER
TESTEG C     R15,@NMALP(R7)
       JEQ   TESTEH
       MOV   @AECRFL(R7),R0
       AI    R0,8
       LI    R2,8
       BL    @APOV
       SETO  @DRAPER
TESTEH INCT  R7
       C     R7,R4
       JEQ   SORNE
       AI    R0,>200
       JMP   TESTE
* test si dernier essai
SORNE  CI    R4,18
       JNE   SORNEA
       B     @ABAND
* mise a jour adresses dans VDP
SORNEA A     @AJ200,@AECRBP
       A     @AJ200,@AECRMP
       MOVB  @C1CADR,@CADREX
       AB    @AJCADR,@CADREY
       LI    R0,>1B00
       LI    R1,CADREY
       LI    R2,2
       BLWP  @VMBW
* test si avant dernier essai
       CI    R4,16
       JNE   SORNEB
       LI    R0,>35B0
       LI    R1,>8000
       LI    R2,64
       BL    @APOV
* vers nouvel essai
SORNEB B     @NOUVES
*************
* MASTER SP1 *
*************
* Routine pour ecrire
* plusieurs octets en
* memoire vive video
APOV   BLWP  @VSBW
       JMP   APOV1
APOV2  MOVB  R1,@>8C00
APOV1  DEC   R2
       JNE   APOV2
       RT
* routine pour retour
* a la mire
* (programme moniteur)
EXIT   CLR   R0
       MOVB  R0,@ETAT
       MOV   @ADRRET,R11
       RT
**********************
* sous programme de  *
* trace de lignes    *
**********************
LIGNE  DATA  REGSP
       DATA  $+2
       C     @REG6,@REG8
       JNE   LIGNEA
       MOV   @REG7,R10
LIGNEC MOV   R10,R1
       MOV   @REG6,R0
       BL    @POINT
       C     R10,@REG9
       JEQ   LIGNEB
       INC   R10
       JMP   LIGNEC
LIGNEA MOV   @REG6,R9
LIGNED MOV   R9,R0
       MOV   @REG7,R1
       BL    @POINT
       C     R9,@REG8
       JEQ   LIGNEB
       INC   R9
       JMP   LIGNED
LIGNEB RTWP
*********************
* sous programme    *
* pour afficher un  *
* point sur l'ecran *
*********************
POINT  MOV   R1,R4
       SLA   R4,5
       SOC   R1,R4
       ANDI  R4,>FF07
```

```
       A    R0,R4
       ANDI R0,7
       S    R0,R4
       LI   R3,>8000
       SRC  R3,0
       MOV  R4,R0
       SWPB R0
       MOVB R0,@>8C02
       SWPB R0
       MOVB R0,@>8C02
       NOP
       MOVB @>8800,R1
       SOC  R3,R1
       SWPB R0
       MOVB R0,@>8C02
       SWPB R0
       SOC  @H4000,R0
       MOVB R0,@>8C02
       NOP
       MOVB R1,@>8C00
       AI   R0,>2000
       SWPB R0
       MOVB R0,@>8C02
       SWPB R0
       MOVB R0,@>8C02
       NOP
       MOVB @REG5,@>8C00
       RT
* sous programme
* affichage des "cases"
SCAR   LI   R7,8
       LI   R9,15
SCAR1  BLWP @LIGNE
       CI   R7,152
       JEQ  SCAR2
       AI   R7,16
       AI   R9,16
       JMP  SCAR1
SCAR2  COC  @H0007,R6
       JNE  SCAR3
       AI   R6,9
       MOV  R6,R8
       JMP  SCAR
SCAR3  AI   R6,7
       MOV  R6,R8
       DEC  R0
       JNE  SCAR
       RT
*********************
* sous programme    *
* pour generer un   *
* nombre aleatoire  *
*********************
SPALEA LI   R4,28645
       MPY  @NOMBAL,R4
       AI   R5,31417
       MOV  R5,@NOMBAL
       RT
* sous programme
* effacement des
* fleches pour erreur
* de logique
EFFLE  MOV  R11,R13
       LI   R7,9
       LI   R0,>2190
       CLR  R1
EFFLE1 LI   R2,16
       BL   @APOV
       AI   R0,>200
       DEC  R7
       JNE  EFFLE1
       LI   R0,>3558
       LI   R2,72
       BL   @APOV
       LI   R0,>3660
       LI   R2,56
       BL   @APOV
       MOV  R13,R11
       RT
* deplacement du
* cadre vers la
* gauche
GAUCH  MOV  R5,R5
       JEQ  GAUCH3
       DEC  R5
       DECT R10
       AI   R12,-16
       SB   @AJCADR,@CADREX
GAUCH2 MOVB @CADREX,R1
       LI   R0,>1B01
       BLWP @VSBW
GAUCH3 B    @CLAV2
* deplacement du
* cadre vers la
* droite
DROIT  CI   R5,4
       JEQ  GAUCH3
       INC  R5
       INCT R10
       AI   R12,16
       AB   @AJCADR,@CADREX
       JMP  GAUCH2
* effacement d'une
* couleur avec
* deplacement vers
* la droite
EFFAC  CLR  R1
       MOV  R12,R0
       LI   R2,8
       BL   @APOV
       MOV  R5,R0
       LI   R7,>8000
       SRC  R7,0
       SZC  R7,@DRAPPO
       JMP  DROIT
* affichage de la
* solution
ABAND  LI   R0,>3508
       CLR  R3
ABAND1 MOV  @STSOL(R3),R8
       MOVB @COULR(R8),R1
       LI   R2,8
       BL   @APOV
       AI   R0,16
       INCT R3
       CI   R3,10
       JLT  ABAND1
       LI   R0,>3608
       CLR  R3
ABAND2 MOV  @STSOL(R3),R8
       MOVB @COULR(R8),R1
       LI   R2,8
       BL   @APOV
       AI   R0,16
       INCT R3
       CI   R3,10
       JLT  ABAND2
       LI   R0,>1558
       LI   R1,SH
       LI   R2,72
       BLWP @VMBW
       LI   R0,>1658
       LI   R1,SB
       BLWP @VMBW
       LI   R0,>3558
       LI   R1,>6000
       BL   @APOV
       LI   R0,>3658
       LI   R2,72
       BL   @APOV
       LI   R0,>15B0
       LI   R1,R
       LI   R2,56
       BLWP @VMBW
       LI   R0,>16B0
       LI   R1,OB
```

```
        LI    R2,64
        BLWP  @VMBW
        LI    R0,>35B0
        LI    R1,>1000
        LI    R2,56
        BL    @APOV
        LI    R0,>36B0
        LI    R2,64
        BL    @APOV
        MOVB  @C1CADR,@DCADRE+1
        MOVB  @C1CADR,@CADREX
        LI    R0,>0400
        MOVB  R0,@DCADRE
        MOVB  R0,@CADREY
        CLR   @DRAPER
        LI    R0,>01A8
        MOV   R0,@AECRMP
        LI    R0,>1B00
        LI    R1,>D000
        BLWP  @VSBW
* scrutation du clavier
ABAND5  LI    R5,>00FF
ABAND3  BLWP  @KSCAN
        DEC   R5
        JEQ   ABANDA
        MOVB  @ETAT,R0
        JEQ   ABAND3
* si touche BACK pressee
* retour au titre
        CB    @RTCLAV,@TBACK
        JNE   ABAND4
        B     @BACK2
* si touche REDO pressee
* nouvelle partie de
* meme niveau
ABAND4  CB    @RTCLAV,@TREDO
        JNE   ABAND3
        B     @REDO
* si la solution a ete
* trouvee, les couleurs
* placees sur la ligne de
* l'essai clignotent
ABANDA  MOV   R15,R15
        JLT   ABANDB
        JMP   ABAND3
ABANDB  MOV   @AECRES(R4),R0
        MOV   R6,R6
        JEQ   ABANDC
        CLR   R6
        CLR   R7
ABANDD  MOV   @STSOL(R7),R8
        MOVB  @COULR(R8),R1
        LI    R2,8
        BL    @APOV
        AI    R0,16
        INCT  R7
        CI    R7,10
        JLT   ABANDD
        MOV   @AECRBP,R0
        AI    R0,>2000
        LI    R1,>D000
        LI    R2,40
        BL    @APOV
        JMP   ABAND5
ABANDC  SETO  R6
        LI    R7,5
        CLR   R1
ABANDE  LI    R2,8
        BL    @APOV
        AI    R0,16
        DEC   R7
        JNE   ABANDE
        MOV   @AECRBP,R0
        AI    R0,>2000
        LI    R1,>8000
        LI    R2,40
        BL    @APOV
* vers scrutation clavier
        JMP   ABAND5
```

Du nouveau sur les cassettes

Gérard Michel

Les programmes sont maintenant enregistrés sur les deux faces de la cassette. Les commentaires parlés nous semblent, après six mois d'utilisation, d'un intérêt relatif. C'est pourquoi nous avons décidé de les supprimer et d'insérer un "catalogue" au début de chaque cassette. Il donne une explication sommaire des programmes ainsi que les langages utilisés. Nous avons prévu des emplacements pour ajouter les numéros de compteur du magnétophone. Puisque ces numéros varient dans des proportions importantes d'un appareil à l'autre, nous les avons remplacés par "XXX", qu'il vous appartiendra de modifier selon les indications fournies par le compteur de votre magnétophone. Nous espérons que ce nouveau système vous conviendra. Les catalogues sont écrits en Basic TI afin qu'ils soient utilisables par tous.

Le magnétophone à cassettes Texas Instruments "Program Recorder"

Cet appareil de construction robuste peut certainement être placé entre des mains peu délicates et convient donc très bien à une utilisation familiale. Contrairement à ce que son nom semblerait vouloir indiquer, il est possible de l'utiliser comme un magnétophone classique puisque tous les réglages habituels sont présents. Les prises destinées à recevoir les trois fiches du câble de connexion sont de mêmes couleurs que celles-ci ; ce n'est pas grand chose, mais cela facilite un peu la vie du programmeur. Ce magnétophone peut recevoir trois types d'alimentation : quatre piles de 1,5 volts, secteur (220 V) ou transformateur 6 volts. Nous utilisons cet appareil de façon intensive depuis environ trois mois sans rencontrer le moindre problème, ce qui tendrait à prouver que la fiabilité est bonne (nous ne sommes pas toujours d'une douceur exemplaire avec le matériel !).

Bibliographie

Gérard Santraille

La conduite du TI-99, de Pierre Willard, aux Editions Eyrolles - 192 pages, 85 FF.

C'est selon moi l'ouvrage français le plus complet sur le TI-99. A côté des classiques descriptions des commandes et instructions du BASIC, on trouve dans ce livre de solides explications sur le codage interne des variables et des chaînes de caractères, l'emploi des opérateurs logiques, l'optimisation des programmes, l'organisation interne du TI-99 et sur de nombreux autres points. L'auteur fait un panorama complet des extensions du TI-99 et des différents fichiers utilisables (la majeure partie du chapitre correspondant est consacrée aux fichiers sur cassette).

Pour agrémenter le tout, cet ouvrage se termine par un recueil d'annexes regroupant, souvent sous forme condensée (tableaux...), des informations précieuses comme les instructions du processeur ou les codes ASCII des caractères.

Un livre à conseiller à tous ceux qui veulent en savoir un peu plus sur le fonctionnement du TI-99.

Jeux et Programmes pour l'ordinateur familial de Texas Instruments, de Gerard Ceccaldi, aux Editions Shift - 100 pages, 155 FF.

C'est le premier tome d'une série de trois livres consacrés au TI-99. Il se présente comme un recueil de 50 programmes, tous en TI Basic et de niveaux très inégaux. A côté de jeux désormais classiques comme les dames, biorythmes, pendu, tiercé, loto, etc... se trouvent quelques programmes plus originaux que vous maîtriserez toutefois rapidement. Les listings publiés dans ce livre sont malheureusement dépourvus de commentaires. Du fait de la longueur de certains d'entre eux, une cassette d'accompagnement n'aurait pas été superflue.

Exercices pour TI-99/4A, de Frédéric Levy, aux Editions du PSI - 144 pages, 82 FF.

Ce livre est la suite logique de "A la découverte du TI-99/4A".

Dans la première partie, l'auteur propose un grand nombre d'exercices de difficulté croissante. Tous les problèmes fréquemment rencontrés en programmation sont abordés : affichage, nombres aléatoires, tableaux, classements, tris, chaînes de caractères, graphismes, etc... Chaque exercice proposé est accompagné d'un organigramme qui facilite l'écriture du programme (les plus courageux pourrons ignorer cet organigramme et concevoir le leur).

La deuxième partie de l'ouvrage donne une solution à chaque problème posé. Le lecteur pourra comparer son programme avec celui de l'auteur. De nombreuses explications sont données en regard des listings.

Je crois pouvoir affirmer que tous ceux qui auront lu ce livre (en faisant les exercices) seront capables de comprendre ou d'écrire n'importe quel programme en Basic TI.

Mon TI-99/4A, de Xavier Ceyrat et Claude Marty, au Editions Eyrolles - 144 pages, 75 FF.

Tout d'abord, je dois dire que seul le désir de vous informer m'a permis d'achever la lecture de ce livre. J'avais en effet la curieuse impression de lire le manuel d'utilisation fourni par Texas Instruments. Il en va de même pour certains exemples de programmation.

Signalons toutefois la présence de quelques petits programmes plus originaux (pas toujours très bien structurés).

En conclusion, je ne pense pas que cet ouvrage puisse apporter grand chose à ses lecteurs. Par contre, si vous avez perdu le mode d'emploi de votre TI...

Initiation au langage assembleur du Texas Instruments TI-99/4A, de Denise Amrouche et Roger Didi, aux Editions Shift - 214 pages, 195 FF.

Le volumineux classeur qui accompagne l'Editeur/Assembleur est très complet mais il a aussi deux gros défauts : il est rédigé en anglais et s'adresse à des personnes connaissant déjà un langage d'assemblage.

"Initiation au langage assembleur" permet un apprentissage progressif. Des exemples concrets, comme la simulation de DISPLAY AT, facilitent la compréhension des textes. Une large place est laissée à l'analyse de la mémoire vidéo en général, et au mode Bit-Map en particulier. Certaines routines auraient pu être écrites de façon plus concise, mais il est vrai qu'un programme "optimisé" est souvent plus difficile à comprendre.

Vous trouverez aussi dans ce livre le programme "LINES" (cassette de la mini-mémoire) désassemblé.

L'ouvrage de Denise Amrouche et Roger Didi est le complément indispensable du Manuel de Texas Instruments pour tous ceux qui désirent s'initier aux joies de l'assembleur.

A vous de programmer

Dans le premier numéro de "99 Magazine", nous vous avions proposé d'écrire un programme sur le thème : la carte de France. Nous avons reçu plusieurs propositions et il fallait faire un choix. Celle de Laurent Desobelle a été selectionnée, et son auteur reçoit un abonnement d'un an avec cassettes. Certains problèmes nous ont empêché de publier cette rubrique dans le numéro 2 et nous nous en excusons.

Cette fois, nous vous proposons d'écrire un programme capable de simuler une horloge, une montre, un chronomètre, ou tout instrument ayant un rapport avec le temps.

Tous les coups sont permis : affichage numérique, à aiguilles... Basic TI, Basic Etendu, Assembleur...

Ce programme ne pourra pas être utilisé de façon pratique puisqu'il est impensable de devoir laisser le TI sous tension uniquement pour avoir l'heure, mais il peut constituer un excellent exercice de programmation. Le facteur temps entrant en ligne de compte, une optimisation sera nécessaire.

Les lecteurs intéressés par ce petit jeu devront nous envoyer leurs réalisations avant le 31 janvier. Le meilleur logiciel sera publié dans le prochain numéro et son auteur recevra un abonnement d'un an avec les cassettes d'accompagnement. Bon courage...

Courrier des lecteurs

Alexandre Duback

1 - Etant possesseur d'un TI-99/4A, je désirerais savoir si l'utilisation de celui-ci via la prise péritel ou UHF n'abîmera pas mon téléviseur SECAM couleur ou tout autre téléviseur à court ou long terme.

2 - D'autre part, je voudrais vous féliciter pour les jeux que vous avez publiés et vous encourager à donner plus souvent la "traduction" en Basic TI pour les programmes en Basic Etendu lorsque vous en avez la possibilité.

Laurent Delaveau - 23 rue Paul Lafargue - 91550 Paray-Vieille Poste

1 - Les images générées par le TI-99/4A, ou par tout autre ordinateur, sont sensiblement différentes des images télévisées : le caractère matriciel de l'affichage d'un ordinateur fait que certains points de l'écran sont très souvent sollicités alors que d'autres le sont très peu. De plus, les images issues d'un ordinateur sont généralement plus contrastées que les images classiques. Tout ceci fait que, seulement à très long terme, la couche de phosphore du tube cathodique peut garder la trace de la matrice d'affichage. Je ne pense pas que cela soit de nature à nous faire délaisser nos TI.

2 - Il n'est pas toujours possible de "traduire" les programmes écrits en Basic Etendu puisque le Basic TI ne comporte pas d'instructions propres à générer des lutins. Par contre, il est vrai que certains programmes en Basic Etendu ne le sont que parce que l'auteur du programme possède ce module, et l'utilise systématiquement.

1 - Comme vous l'avez souligné dans l'un de vos articles, les extensions en pépiphériques du TI-99 sont assez onéreuses. Je voudrais savoir si l'on peut s'attendre à une baisse de prix de ce matériel et tout d'abord du boîtier d'extension. J'ai lu qu'une baisse très importante du TI-99 aurait lieu très prochainement (il serait vendu aux alentours de 750 francs), le prix des périphériques va t'il suivre ?

2 - D'autre part, à la suite de l'article "fabriquez vous-même vos joysticks" et pour en revenir au boîtier d'extension, est-il possible d'en bricoler un soi-même ? Pourriez-vous nous dire comment car, en fait, ce fameux boîtier d'extension indispensable et si cher ne me paraît être qu'une "vulgaire boîte à prises".

Patrick Wairy - 4 rue Léon Blum - 02500 Hirson

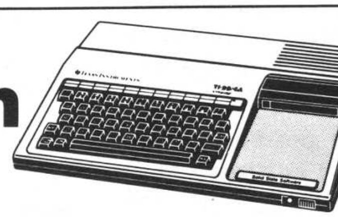
1 - Au sujet des prix, vous avez pu constater que, si le TI-99 n'est pas encore vendu 750 FF, la console et les extensions bénéficient depuis peu de baisses importantes.

2 - Le boîtier d'extension n'est pas seulement une "boîte à prises". Il contient l'alimentation des cartes périphériques (cette alimentation doit être parfaitement stable), une carte d'extension qui permet de faire la liaison entre la console et les périphériques, et une ventilation indispensable au bon fonctionnement du système. Tout ceci est difficilement réalisable par un amateur et revien-

Ordinateurs familiaux



Vous avez choisi
le champion



TEXAS 099/4A

Tous les logiciels sont disponibles chez GOLDY

LOGICIELS DE JEUX
CARTOUCHES

Jeux vidéo 1 : 3 jeux d'adresse (safari, Bing-ball, duel)	149 F
Chasse au Wumpus	199 F
Football	199 F
Jeux Vidéo II : 2 Jeux de réflexion et de mémoire	149 F
A-MAZE-ING : le chat et la souris	199 F
ATTACK	199 F
BLASTO : combat de chars	199 F
BLACK-JACK : pocker*	199 F
HUSTLE : jeu de poursuite	199 F
ZERO-ZAPP : Texas-flipper	149 F
HANGMANN : jeu du pendu	199 F
CONNECT-4 : jeux du morpion	199 F
YAHTZEE : jeux de Yam	199 F
TUNNELS OF DOOMS : jeux de type Adventure avec graphique*	249 F
TOMBSTONE CITY : far-west du XXIe siècle	199 F
TI-INVADERS : envahisseurs de l'espace	199 F
CAR WARS : course de voitures	199 F
ALPINER : utilisation possible du PHP 1500	249 F
MUNCHMAN : jeu du glouton	249 F
OTHELLO	249 F
CHISHOLM TRAIL	249 F
PARSEC : utilisation possible du PHP 1500	249 F

LOGICIELS DE JEUX
CASSETTES

Jeux Rétro I : 4 jeux, Biorythme, reconstitution de mots	75 F
Jeux Rétro II : 4 jeux	75 F
PIRATE Adventure + module 3041 T/US	249 F
ADVENTURELAND (3)	149 F
MISSION IMPOSSIBLE (3)	149 F
VOODOO CASTLE (3)	149 F
THE COUNT (3)	149 F
STRANGE ODYSSEY (3)	149 F
MYSTERY FUN HOUSE (3)	149 F
PYRAMID OF DOOM (3)	149 F
GHOST TOWN (3)	149 F
SAVAGE ISLAND I & II (3)	149 F
GOLDEN VOYAGE (3)	149 F

LOGICIELS ÉDUCATIFS
CARTOUCHES

PRESCHOOL, early learning fun*	149 F
BEGINNING GRAMMAR*	149 F
MAGIE DES NOMBRES	149 F
ECHECS : apprentissage et jeux	375 F
EARLY READING* (1)	249 F
MUSIC MAKER (Initiation et perfectionnement au solfège)	249 F
ADDITION- SUBSTRACTION I*	249 F
ADDITION- SUBSTRACTION II*	249 F
MULTIPLICATION I*	249 F
DIVISION*	249 F
CROCODILE SAVANT	199 F
ADDITION CANON	199 F
DIVISION DEMOLITION	199 F
DRAGON SAVANT	199 F
MISSION MOINS	199 F
MÉTÉOR MULTIPLICATION	199 F

CASSETTES

LE BASIC étendu par soi-même (2)	75 F
LE BASIC par soi-même	75 F
PONCTUATION	119 F
NE CONFONDEZ PAS	119 F
DIVISEURS PGCD, PPCM	119 F

LOGICIELS GESTION PERSONNELLE
CARTOUCHES

GESTION DE FICHIERS	375 F
GESTION DE RAPPORTS	375 F
FICHIER D'ADRESSES	775 F
GESTION PRIVÉE (L'expansion)	375 F
TI-CALC	375 F
CASSETTE Conseil financier	75 F

AUTRES LOGICIELS
CARTOUCHES

GYMNASTIQUE*	149 F
SPEECH EDITOR* (1)	199 F
STATISTICS*	250 F
EXTENDED-BASIC	599 F
TERMINAL EMULATOR II : permet de communiquer par téléphone si utilisé avec un modem et une carte RS 232 (PHP 120). Avec le PHP 1500 permet de prononcer en anglais des phrases composées sur l'écran	665 F
ÉDITOR-ASSEMBLER	599 F
MINI-MÉMOIRE : permet de stocker 5 Koctets et donne un accès simplifié au langage Assembleur par l'intermédiaire d'une cassette	599 F

Tous ces logiciels sont livrés en français, sauf ceux marqués*

1) nécessite le PHP 1500
2) nécessite le module PHM 3026
3) nécessite le module 3041 T/US

GARANTIE GOLDY
Tous ces logiciels sont garantis 1 an pièces et main-d'œuvre. Si par extraordinaire, ils présentaient un défaut quelconque ils sont immédiatement échangés.
ENVOI EN RECOMMANDÉ

BON DE COMMANDE OU DE DOCUMENTATION
à renvoyer à GOLDY
BP 274 - 27002 ÉVREUX Cédex

☐ Envoyez-moi les logiciels indiqués ci-dessous :
________ Prix ________
________ Prix ________
________ Prix ________
________ Prix ________

Participation aux frais d'envoi :
EN RECOMMANDÉ 19,00 F
TOTAL : F

☐ Je vous règle par chèque bancaire, CCP 3 volets ou mandat-lettre à l'ordre de GOLDY.

☐ Je désire recevoir une documentation en couleurs sur le TEXAS 099/4A, ses logiciels et ses accessoires. **194 TI/108**

NOM ________
Prénom ________
N° ____ Rue ________

Code postal ________
VILLE ________

drait, construit à une seule unité, à plus de 4000 FF (le boîtier est actuellement vendu moins de 1000 FF).

1 - Peut-on déplacer le curseur de droite à gauche (je souhaite réaliser pour ma classe un programme d'opérations en colonnes à plus d'un chiffre) ?

2 - Quelles sont ces minuscules dont vous parlez page 8 du N° 1 dans l'article sur "les points forts du TI-99". Des minuscules me rendraient bien service pour l'apprentissage de la lecture.

3 - Afin d'utiliser ma console à moindre frais avec mes élèves, je vais me résoudre à acheter un poste TV noir et blanc ainsi que l'interface qui doit pallier l'absence de prise Péritel. Est-ce que cela vous paraît raisonnable ?

4 - Enfin, je voudrais savoir si les logiciels existants ainsi que les périphériques du TI-99/4A seront compatibles avec le nouvel ordinateur Texas Instruments.

Claude Ruetsch - 14 Avenue des Oiseaux de mer - 17140 Lagord

1 - Il est toujours possible de déplacer un curseur de droite à gauche : avec la "flèche à gauche" du clavier (FCTN-S) en mode commande, ou par programme, où tout est possible. L'article de Roger Didi "Apprendre en programmant" vous donne des indications à ce sujet.

2 - Le TI-99/4A ne possède pas de vraies minuscules. La touche ALPHA LOCK, si elle est relâchée, permet seulement d'obtenir des "petites majuscules". Par contre, vous pouvez très bien programmer votre propre jeu de minuscules. Gérard Santraille, dans la rubrique "trucs et astuces", vous indique la marche à suivre pour parvenir à ce résultat.

3 - Le prix d'un téléviseur noir et blanc augmenté de celui d'une interface SECAM ne doit pas tellement s'éloigner du prix d'un récepteur couleur de bas de gamme. J'utilise un poste couleur valant environ 2500 FF, et qui me donne entière satisfaction. De plus, les 16 couleurs du TI constituent une de ses plus grandes qualités, et il semble que cela soit un atout majeur lorsque l'on veut intéresser les enfants. Mon raisonnement n'étant pas forcément valable (je ne suis pas enseignant), je pense que vous devriez en parler à d'autres professeurs.

4 - Vous faites certainement allusion au TI-99/8, annoncé il y a quelques temps par Texas Instruments. Les logiciels et périphériques du TI-99/4A auraient dû être compatibles avec ce nouvel ordinateur. Malheureusement, il ne sortira certainement jamais puisque Texas U.S. a décidé d'imterrompre la fabrication des ordinateurs familiaux. Mais, sait-on jamais...

La cassette 99B que j'ai reçu en septembre a une erreur dans le programme CHARDEF car, à l'exécution du programme, le TI-99/4A affiche FOR-NEXT ERROR. Je voudrais savoir ce que je pourrais faire pour la supprimer.

Jean-Claude Galé - 1, cité Foch, Boulevard Claude Debussy-65000 Tarbes

D'une manière générale, lorsqu'un programme provoque l'affichage de "FOR-NEXT ERROR" sans numéro de ligne, et que la commande "LIST" fait apparaître des symboles bizarres, il s'agit d'un programme écrit en Basic Etendu exécuté sans le module nécessaire.



Une suggestion : le Basic Etendu étant un complément que beaucoup achètent après quelque temps, serait-il possible de proposer une rubrique permettant d'exploiter peu à peu toutes ses possibilités ? (Des petits programmes progressifs comme ceux de Monsieur Didi dans le N° 1 seraient les bienvenus). Le manuel entièrement en anglais n'est pas très attrayant !

Jean-Edouard Bisiaux - 8 Résidence Sainte Anne - Locon - 62400 Béthune

Nous pensons commencer une série d'articles sur les particularités du Basic Etendu dès le prochain numéro, car il semble que le manuel pose des problèmes à de nombreux lecteurs. Nous sommes heureux de constater que les articles de Roger Didi vous plaisent.

Dans la revue, les programmes en Basic Etendu devrait être indiqués très clairement.

Jacques Lorig - 5 Avenue de Valdonne - 13013 Marseille

Nous sommes entièrement d'accord avec vous. Dès ce numéro, les langages sont clairement indiqués dans les articles d'accompagnement ainsi qu'en tête de chaque listing.

La cassette N° 1 de "99 Magazine" ne fonctionne pas. Pour le programme "Alphabet", j'obtiens le message d'erreur BAD SUBSCRIPT IN 730, et BAD VALUE IN 220 pour "Le dernier robot".

Chantal Ralaimihoatra - 13 rue de l'Epée de bois - 75005 Paris

Vos problèmes proviennent certainement du fait que vous essayez de faire "tourner" ces programmes, écrits en Basic TI, en Basic Etendu. Un programme écrit en Basic TI ne fonctionne pas forcément en Basic Etendu pour les raisons suivantes :

- le jeu de caractères du Basic Etendu est, contrairement à ce que l'on pourrait croire, moins étendu que celui du Basic TI (codes ASCII de 30 à 143 contre 32 à 159 pour ce dernier) ;
- la mémoire utilisable en Basic Etendu est plus restreinte qu'en Basic TI (864 octets de moins), sauf si vous utiliser l'extension 32 K ;
- un programme écrit en Basic TI peut comporter une variable ayant un nom correspondant à un mot clé du Basic Etendu.

Quand j'ai vu "99 Magazine" dans un kiosque, j'ai littéralement sauté sur la chose. Je l'ai dévorée en une journée (et un peu de nuit) et j'ai commencé dès le lendemain à copier les programmes listés dans le magazine. J'ai aimé la présentation et ces programmes sont amusants au possible. En relisant l'éditorial pour une troisième fois, je me suis décidé à vous écrire.

Une seule ombre au tableau : les programmes tels "Le dernier robot" et "Harmonium" sont écrits tellement petits que je me suis trompé de ligne assez souvent. Mais, je comprends que, pour des raisons d'espace, il est impossible de les écrire plus gros. Donc, je ne vous en veux pas du tout.

J'espère que cette lettre d'un lecteur outre-Atlantique vous a fait plaisir. Longue vie à "99 Magazine" et, comme on dit chez nous : "lâchez pas, les gars !" (Mes excuses à Catherine et à Denise, nous n'avons pas encore féminisé cette expression.

Stéphane Poudret - C.P. 386, Deschaillons, Comté Lotbinière (Québec), GOS 1GO - Canada

Votre lettre nous fait effectivement plaisir, cher "cousin", et ne vous inquiétez pas, nous ne "lâcherons pas".

Erratum

Nous vous proposons ci-dessous un erratum des deux premiers numéros de "99 Magazine". Nous vous prions de bien vouloir excuser ces péchés de jeunesse ; notre rédaction a eu quelques difficultés initiales avec son rédacteur principal. Elle est maintenant réorganisée et nous pouvons vous assurer que la vérification des articles et programmes a été rendue beaucoup plus rigoureuse que par le passé.

Numéro 1 de 99 Magazine

L'article Fichiers en BASIC du numéro 1 a été réalisé conjointement par Denise Amrouche et Simon Tchek.

Numéro 2 de 99 Magazine

Page 7 : Micro-Monde LOGO - Le TI-99/4A n'est pas le seul ordinateur individuel capable de générer des lutins : le Commodore 64 offre une possibilité similaire.
De plus, les lutins du Basic Etendu n'ont pas de forme prédéfinie, sinon la forme des caractères ASCII correspondant aux codes employés pour l'instruction CALL SPRITE.

Page 18 : Jeu du Simon - Il faut évidemment remplacer PRINT AT par DISPLAY AT.

Page 41 : La sauvegarde des fichiers et programmes - Il est parfaitement possible de lire les annonces sonores après avoir écrit "OLD CS1" et même après avoir appuyé sur "ENTER" puisque, lorsque l'ordinateur affiche "REWIND CASSETTE TAPE THEN PRESS ENTER", le défilement de la bande est toujours possible.
La télécommande sur les deux unités est possible sans problème avec l'ancien câble à deux dérivations.

Page 52 : Accélérez vos programmes en Basic Etendu - Ce truc ne fonctionne pas (hélas !) pour plusieurs raisons :

- l'instruction CALL LOAD ne peut fonctionner qu'avec une extension de mémoire ;
- elle ne peut pas être placée en tête de programme puisqu'elle doit être précédée par CALL INIT
- l'adresse −31878 (837A en hexadécimal) n'est pas l'adresse où se fait le guidage de tous les sprites mais l'adresse qui contient le nombre de lutins qui peuvent être en déplacement. Il est évident qu'une seule adresse (octet) ne peut pas contenir un système aussi complexe ;
- ce "truc" ne sert à rien car cette adresse est automatiquement mise à jour par l'interpréteur (il suffit de faire un CALL PEEK à cette même adresse, lorsque l'on utilise des lutins, pour s'en rendre compte).

Cassette A et B

Le programme CHARDEF n'est pas en Basic simple comme annoncé à la page 4 de la revue, mais en Basic Etendu. En outre,le programme "Division" se trouve sur la cassette 99B et non sur la cassette 99A.

Cassette C

Programme DAMES : ce programme est écrit en Basic Etendu et non en Basic simple comme l'indique l'annonce sonore.

Cassette 2

Programme Canon de Purcell : L'instruction MERGE n'est pas valide avec les cassettes. Le programme n'est donc pas utilisable. Afin de réparer cette erreur, nous avons décidé de le réenregistrer sur la cassette d'accompagnement de ce numéro.

Les lecteurs qui possèdent une unité de disquettes, et qui n'auraient pas l'intention d'acquérir la cassette 3, peuvent "récupérer" ce programme en employant la méthode suivante :

- charger le programme CANON 1 en mémoire depuis la cassette ;
- sauvegarder ce programme sur disquette avec DSK1.CANON1 ;
- charger le programme CANON 2 en mémoire depuis la cassette ;
- sauvegarder ce programme sur disquette avec DSK1.CANON2, MERGE ;
- charger le programme CANON 3 en mémoire depuis la cassette ;
- sauvegarder ce programme sur disquette avec DSK1.CANON3, MERGE ;
- charger le programme CANON1 en mémoire avec OLD DSK1. CANON1 ;
- charger le programme CANON2 en mémoire avec MERGEDSK1. CANON2 ;
- charger le programme CANON3 en mémoire avec MERGEDSK1. CANON3 ;
- sauvegarder le programme ainsi reconstitué avec SAVE DSK1. CANON ;
- détruire les programmes CANON1, CANON2 et CANON3 (facultatif).

Ceci n'intéresse que les lecteurs ayant reçu la cassette N° 2 avant le 1er décembre : ces cassettes ont été modifiées et le programme "Canon de Purcell" est maintenant utilisable.

Petites annonces (gratuites)

Vends contrôleur de disquettes indépendant (ne nécessite pas le boîtier d'extension) - 1000 FF - S'adresser au journal qui transmettra.

Achète : Basic Etendu + cartouches de jeux-Ecrire à Monsieur Jacques Lorig - 5 Avenue de Valdonne - 13013 Marseille.

Vends TI-99/4A (janvier 82) très bon état + ouvrages sur le TI-99 + 1 poignée de jeu (suivant le plan de "99 Magazine" N° 1). Le tout vendu 1650 FF-TEL : 938-13-55 après 18h S.V.P.

Le club Tigre 13

Nous sommes heureux de vous apprendre la création d'un nouveau club d'informatique à Marseille.

Ce club est principalement axé sur l'étude du TI-99/4A. Les buts poursuivis sont les suivants :

- formation des débutants (initiation) ;
- réunions d'échanges (perfectionnement) : Basic Etendu, Assembleur, élaboration de programmes en commun ;
- libre service : échange de logiciels, d'idées, discussions.

Informations pratiques :

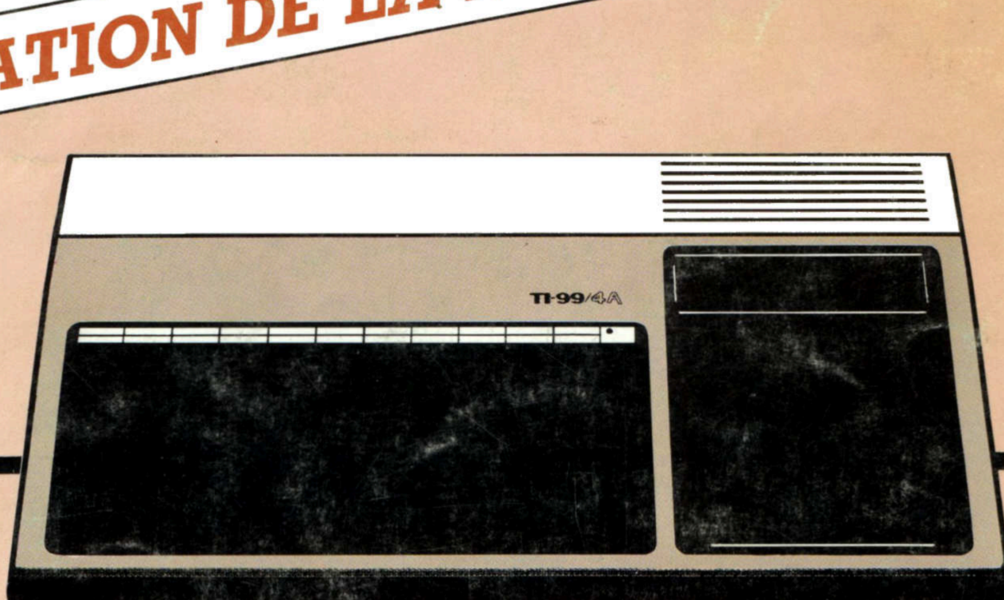
- nous comptons actuellement environ 60 adhérents ;
- nous nous réunissons tous les mercredis et tous les samedis à l'U.A.S. Rouet, Rue Benedetti, 13008 Marseille ;
- horaires du mercredi : 19 h 30 - 21 h 30 (une semaine initiation, une semaine perfectionnement) ;
- horaire du samedi : 14 h - 18 h (libre service) ;
- la cotisation annuelle est fixée à 100 FF et nous demandons cinq enveloppes timbrées avec nom et adresse de l'adhérent.

Pour de plus amples renseignements, joindre Monsieur Malagoli, trésorier, 43 rue de la Loge, 13002 Marseille. Téléphone : (91) 91-40-10.

CASSETTE D'ACCOMPAGNEMENT DISPONIBLE

99 MAGAZINE

GRAPHISME EN HAUTE-RESOLUTION
BASIC, PASCAL, LOGO, ASSEMBLEUR...
DES PROGRAMMES POUR VOTRE TI
UTILISATION DE LA MINI-MEMOIRE



NUMÉRO 4 - Trimestriel - Mars 1984 - 40 F

ISSN 0759-9005

Sommaire

	Page
Editorial par Hervé Thiriez	5
SOS Hélicoptère par Michel Teyssou	6
Utilisation de la Mini-mémoire par Denise Amrouche	8
Communiqué de Texas Instruments France	14
Le serveur de restaurant par Edouard Krok	15
« Echecs » à l'essai par Alain Paloma	18
Combat de chars par Jean-Claude Hoet	19
Réalisez votre cordon de magnétophone par Guy Fix	21
Apprendre le Basic en programmant par Roger Didi	22
Jeu de Mémorisation par Jean-Claude Cornevin	28
Jack-Pot par Marianne Sutz	29
Les nombres en Logo par Sophie	32
Graphisme en haute résolution sur le TI-99 par Gérard Santraille et Jean-Luc Bazanegue	35
Sous-programmes en Assembleur par Jean-Luc Bazanegue	41
Horloge par Christian Monfort	46
Un détecteur de lumière pour le TI-99 par Guy Fix	47

OOOO Le club M.T.I.	47
Le système P-UCSD par Gérard Santraille	48
Les lutins du Basic Etendu par Julien Thomas	51
Chargeur automatique en Basic Etendu par Eric Thomas	56
Le solitaire par Olivier Colin	57
Claustrophobia par Julien Thomas	58
« DISPLAY » et « ACCEPT » avec le module « PRK » par Sophie Ehster	60
Une erreur dans le manuel de la Mini-mémoire par Jean-Luc Bazanegue	61
Petites annonces	61
Réparez votre Assembleur ligne par ligne par Gilles Pavy	62
Courrier des lecteurs par Alexandre Duback	63
A vous de programmer	64
Trucs et astuces	7 - 18 - 21 - 50

Les annonceurs
PSI Diffusion : page 2 - SICOB : page 4 - Sybex : page 67 - L'Ordinateur Individuel : page 68.

Editions MEV - 49, rue Lamartine - 78000 Versailles - Tél. : (3) 951.24.43

(Dépôt légal : 1er trimestre 1984)

Editorial

Ce numéro de « 99 Magazine » est plein à craquer ; il a donc été nécessaire de modifier la mise en page de certains listings afin que tous les articles et programmes puissent être publiés.

Vous êtes nombreux à ne pas pouvoir vous procurer le module « Basic Etendu » ; nous avons donc décidé d'augmenter le nombre de programmes en Basic TI. Cependant, « 99 Magazine » est la revue de TOUS les utilisateurs du TI-99 et les « mordus » de Pascal, Assembleur, Logo ou Basic Etendu trouveront dans ce numéro de quoi occuper leurs loisirs.

En outre, vous êtes de plus en plus nombreux à nous faire le plaisir de lire cette revue, ce qui renforce notre détermination à continuer la publication de « 99 Magazine ».

*Dans ce numéro, vous trouverez de nombreux programmes : **Michel Teyssou** vous transforme en pilote d'hélicoptère, **Edouard Krok** vous propose un stage de serveur de restaurant, pendant que **Jean-Claude Hoet** vous entraine dans un combat de chars.*
***Jean-Claude Cornevin** met votre mémoire visuelle à l'épreuve alors que **Marianne Sutz,** avec « Jack-Pot », met votre fortune en situation périlleuse. **Christian Monfort** transforme votre TI en horloge, **Olivier Colin** vous propose une partie de « Solitaire » et **Julien Thomas** vous présente une nouvelle production du club TISOFT-HOME : « Claustrophobia ».*
*Au chapitre des programmes il faut aussi signaler un chargeur automatique en Basic Etendu, écrit par **Eric Thomas.***

***Denise Amrouche** vous indique comment utiliser la Mini-mémoire sans savoir programmer en Assembleur pendant que **Roger Didi** poursuit sa série d'articles sur l'apprentissage du Basic.*
***Gérard Santraille** et **Jean-Luc Bazanegue** implantent de nouvelles fonctions graphiques sur votre TI à l'aide de routines en Assembleur. **Gérard** vous propose aussi un article sur le Système P-UCSD alors que **Jean-Luc** vous explique le fonctionnement des sous-programmes en Assembleur.*
*En outre, **Julien Thomas** débute, avec les lutins, une série d'articles sur les particularités du Basic Etendu pendant que **Sophie** vous fait découvrir comment Logo manipule les nombres.*

*Pour les bricoleurs : **Guy Fix** vous communique des renseignements qui vous permettront de réaliser vous-même un cordon de magnétophone ou de transformer votre TI en détecteur de lumière. Vous trouverez aussi dans ce numéro plusieurs trucs et astuces.*

Hervé Thiriez

Rédacteur en chef - Directeur de la publication : Hervé Thiriez. **Comité de rédaction :** Jean-Luc Bazanegue, Thomas Coppens, Roger Didi, Gérard Santraille. **Dessins :** Laurent Bidot. **Ont collaboré à ce numéro :** Denise Amrouche, Jean-Luc Bazanegue, Olivier Colin, Jean-Claude Cornevin, Roger Didi, Alexandre Duback, Sophie Ehster, Guy Fix, Jean-Claude Hoet, Edouard Krok, Christian Monfort, Alain Paloma, Gilles Pavy, Gérard Santraille, Sophie, Marianne Sutz, Michel Teyssou, Robert Thibodeau, Eric Thomas, Julien Thomas, Eric Woerner.

Editions MEV - 49, rue Lamartine - 78000 Versailles. Tél. : (3) 951.24.43.

Régie publicitaire : Force 7, Anne Jourdan, 39, rue de la Grande-aux-Belles, 75483 Paris Cedex 10. Tél. : (1) 238.66.10.

Diffusion auprès des boutiques et libraires : Editions du PSI, 44-51, rue Jacquard, BP 86, 77400 Lagny-sur-Marne.

Composition : Télécompo, 13-15, avenue du Petit Parc, 94300 Vincennes.

Impression : Imprimerie Rosay, 47, avenue de Paris, 94300 Vincennes. Tél. : (1) 328.18.63.

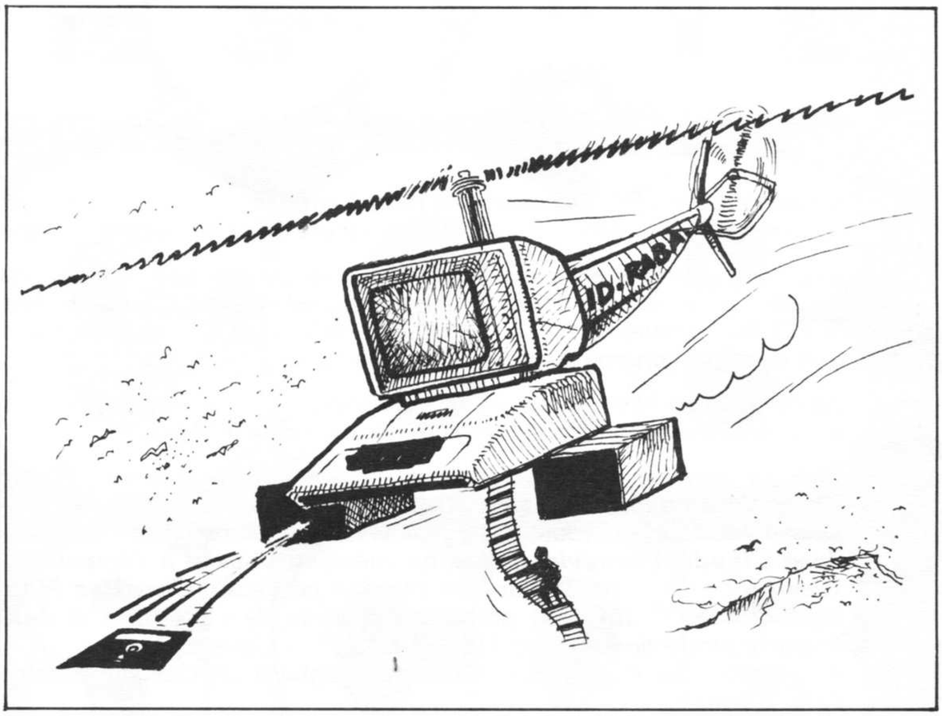
S.O.S Hélicoptère

Michel Teyssou

Trois alpinistes sont bloqués par le mauvais temps au sommet d'une montagne. Votre mission consiste à aller les récupérer avec un hélicoptère. Vous devez donc partir de la vallée et vous diriger vers le sommet de la montagne, sur lequel vous devrez vous poser. Pour diriger votre appareil, utilisez les touches fléchées.

L'hélicoptère n'est pas d'un modèle très récent et les commandes ne réagissent pas toujours très bien; une parfaite maîtrise de l'appareil demande un peu d'entraînement. De plus, la visibilité est très mauvaise : des nuages défilent constamment sur la montagne.

Lorsque vous aurez récupéré un alpiniste (l'hélicoptère n'accepte qu'un passager), il faudra le ramener dans la vallée, si possible en bon état.
Un petit conseil : si vous avez les nerfs fragiles, n'utilisez pas ce programme écrit en **Basic Etendu**. ■



```
100 ! ****************
110 ! *  SOS HELICO  *
120 ! ****************
130 ! * BASIC ETENDU *
140 ! ****************
150 ! *  COPYRIGHT   *
160 ! *MICHEL TEYSSOU*
170 ! *      ET      *
180 ! * 99  MAGAZINE *
190 ! ****************
200 DATA 392,1,392,3,392,1,523,4,523,4,58
7,4,587,4,784,7,659,2,523,4,392,30 !MUS
IQUE
210 CALL CLEAR
220 CALL SCREEN(5)
230 CALL COLOR(1,2,2)
240 GOSUB 2960
250 CALL COLOR(14,16,2)
260 A$=RPT$("0",12)&"02000000A14FA1"&RPT$
("0",18)&"AA0070C8E4F2BECC78FC" !HELICO
270 AA$=RPT$("0",12)&"AA000E13274F7D331E3
F"&RPT$("0",12)&"A000000082F582" !HELIC
O
280 B$=RPT$("0",12)&"0500000041AF41"&RPT$
("0",18)&"550070C8E4F2BECC78FC" !HELICO
290 BB$=RPT$("0",12)&"55000E13274F7D331E3
F"&RPT$("0",12)&"4000000085F285" !HELIC
O
300 C$="99423C1818181818" ! BONHOMME RAME
NE
310 D$="42427E4242427E42" ! ECHELLE
320 T$="18187E18183C7E7E" ! TOMBE
330 E$="0102040810204080" ! DESSIN
340 F$="8040201008040201" ! DESSIN
350 G$=RPT$("01",8)!DESSIN
360 H$=RPT$("80",8)!DESSIN
370 I$="FFFF424242FFFF" !   BASES
380 X$="000F3F7FFFFF3FBF9FC3FFFF7F3F0F000
0E0F8FEFFFFFFFFFFEF8F0F0E0C080" !NUAGES
390 Y$="8903271F9E1C39F83F1E9C1D270389002
280C87072F0F83E3870F2F0C88022" !EXPLOSI
ON
400 J$="0000010100070B112001020202020206000
000808000E0D0880480404040404060" !ALPINIS
TE
410 K$="0000211108070301000102020202020206000
000848810E0C0800080404040404060" !ALPINIS
TE
420 L$="0000010100472B11010F080818000000
0088888810E0C0808080404040404060" !ALPINIS
TE
430 M$="001011111180703010101020202020206000
000808000E2D48880F0101018" !ALPINISTE
440 CALL DELSPRITE(ALL)
450 FOR I=2 TO 11 :: CALL COLOR(I,16,2)::
 NEXT I
460 CALL COLOR(12,12,2)
470 CALL CHAR(56,BB$)!HELICO
480 CALL CHAR(60,AA$)!HELICO
490 CALL CHAR(136,A$)!HELICO
500 CALL CHAR(140,B$)!HELICO
510 CALL CHAR(128,J$)!ALP.
520 CALL CHAR(132,K$)!ALP.
530 CALL CHAR(120,E$)!DESSIN
540 CALL CHAR(121,F$)!DESSIN
550 CALL CHAR(123,G$)!DESSIN
560 CALL CHAR(122,H$)!DESSIN
570 CALL CHAR(125,I$)!BASES
580 CALL CHAR(126,D$)!ECH.
590 CALL CHAR(124,C$)!BONH.
600 CALL CHAR(127,T$)!TOMBE
610 CALL CHAR(92,X$)!NUAGES
620 CALL CHAR(44,Y$)!EXPLO.
630 CALL CHAR(36,L$)!ALP.
640 CALL CHAR(40,M$)!ALP.
650 CALL HCHAR(23,5,125)
660 CALL HCHAR(5,30,125)
670 CALL SPRITE(#10,140,16,160,26)
680 CALL MAGNIFY(3)
690 !--AFFICHAGE DU DESSIN--
700 CALL HCHAR(5,29,120)
710 CALL HCHAR(6,32,121)
720 CALL HCHAR(7,27,123)
730 CALL HCHAR(10,25,123)
740 CALL HCHAR(12,24,123)
750 CALL HCHAR(7,29,123)
760 CALL HCHAR(5,31,121)
770 CALL HCHAR(10,27,123)
780 CALL HCHAR(15,14,123)
790 CALL HCHAR(19,7,123)
800 CALL HCHAR(21,6,123)
810 CALL HCHAR(24,29,121)
820 CALL HCHAR(22,9,123)
830 CALL HCHAR(22,28,122)
840 CALL HCHAR(21,28,122)
850 CALL HCHAR(16,24,122)
860 CALL HCHAR(23,28,121)
870 CALL HCHAR(24,3,121)
880 CALL HCHAR(17,23,122)
890 CALL HCHAR(20,23,122)
900 CALL HCHAR(22,14,122)
910 CALL HCHAR(23,25,122)
920 CALL HCHAR(22,24,122)
930 CALL HCHAR(13,16,120)
940 CALL HCHAR(20,27,121)
950 CALL HCHAR(19,26,121)
960 CALL HCHAR(18,25,121)
970 CALL HCHAR(17,24,121)
980 CALL HCHAR(14,15,120)
990 CALL HCHAR(15,23,121)
1000 CALL HCHAR(14,22,121)
1010 CALL HCHAR(13,21,121)
1020 CALL HCHAR(12,20,121)
1030 CALL HCHAR(16,14,120)
1040 CALL HCHAR(11,19,121)
1050 CALL HCHAR(17,12,121)
1060 CALL HCHAR(16,11,121)
1070 CALL HCHAR(20,12,121)
1080 CALL HCHAR(17,13,120)
1090 CALL HCHAR(21,13,121)
1100 CALL HCHAR(23,14,121)
1110 CALL HCHAR(24,15,121)
1120 CALL HCHAR(16,22,121)
1130 CALL HCHAR(6,28,120)
1140 CALL HCHAR(18,23,121)
1150 CALL HCHAR(19,22,121)
1160 CALL HCHAR(21,23,121)
1170 CALL HCHAR(22,24,121)
1180 CALL HCHAR(16,10,120)
1190 CALL HCHAR(24,25,121)
1200 CALL HCHAR(17,9,120)
1210 CALL HCHAR(18,8,120)
1220 CALL HCHAR(20,7,120)
1230 CALL HCHAR(8,27,120)
1240 CALL HCHAR(22,6,120)
1250 CALL HCHAR(24,4,120)
1260 CALL HCHAR(24,2,120)
1270 CALL HCHAR(23,9,120)
1280 CALL HCHAR(9,26,120)
1290 CALL HCHAR(21,10,120)
1300 CALL HCHAR(20,11,120)
1310 CALL HCHAR(8,29,120)
1320 CALL HCHAR(9,28,120)
1330 CALL HCHAR(11,25,120)
1340 CALL HCHAR(12,26,120)
1350 CALL HCHAR(14,25,120)
1360 CALL HCHAR(13,24,120)
1370 CALL HCHAR(14,23,120)
1380 CALL HCHAR(11,18,120)
1390 CALL HCHAR(12,17,120)
1400 V=15 :: RANDOMIZE :: GOSUB 1820 :: D
ISPLAY AT(24,3)SIZE(5):"12345"
1410 DISPLAY AT(6,1)SIZE(17)
1420 DISPLAY AT(8,1)SIZE(14)
1430 !--TEST DES NUAGES-----
1440 CALL COINC(ALL,C):: IF C=-1 THEN CAL
```

```
L DELSPRITE(ALL):: GOTO 1400
1450 CALL SPRITE(#10,140,16,160,26)
1460 !---1er TABLEAU--------
1470 CALL KEY(0,K1,S)
1480 CALL COINC(ALL,C):: IF C=-1 THEN CAL
L COLOR(#10,1):: T=1
1490 T=T+1 :: IF T=5+TT THEN CALL COLOR(#
10,16):: T=0
1500 CALL POSITION(#10,X,Y)
1510 IF Y>30 AND Y<70 THEN IF X+Y>173 AND
 X+Y<183 THEN 1920
1520 IF Y<134 AND Y>20 THEN IF X+Y>200 AN
D X+Y<212 THEN 1920
1530 IF Y>134 AND Y<250 THEN IF Y-X>67 AN
D Y-X<77 THEN 1920
1540 IF Y>166 AND Y<220 THEN IF Y+X*5/7>2
37 AND Y+X*5/7<243 THEN 1920
1550 IF Y>250 THEN 1920
1560 IF X>180 AND X<250 THEN 1920
1570 CALL COINC(ALL,C):: IF C=-1 THEN CAL
L COLOR(#10,1):: T=1
1580 IF K1=68 OR K1=69 OR K1=88 OR K1=83
THEN 1680
1590 CALL SOUND(60,-8,15,200,30,200,30,50
00,30)
1600 CALL SOUND(50,110,12)
1610 CALL POSITION(#10,X,Y)
1620 IF X>152 AND X<165 THEN IF Y<32 AND
Y>20 THEN CALL MOTION(#10,0,0):: CALL L
OCATE(#10,160,26):: IF IPO=1 THEN 2650 EL
SE 1770
1630 IF IPO=1 AND X=17 AND Y=225 THEN 177
0
1640 IF IPO=1 THEN 1660
1650 IF X>12 AND X<23 THEN IF Y<232 AND Y
>217 THEN CALL DELSPRITE(#10):: CALL HC
HAR(3,29,140):: CALL HCHAR(3,30,142):: CA
LL HCHAR(4,30,143):: CALL HCHAR(4,29,14
1):: GOTO 2110
1660 CALL MOTION(#10,5,0)
1670 GOTO 1770
1680 IF K1=69 THEN CALL MOTION(#10,-8,0):
: GOTO 1720
1690 IF K1=68 THEN CALL MOTION(#10,0,8)::
 GOTO 1720
1700 IF K1=88 THEN CALL MOTION(#10,8,0)::
 GOTO 1720
1710 IF K1=83 THEN CALL MOTION(#10,0,-8)
1720 CALL SOUND(-50,-5,6,120,10)
1730 IF K1<>83 THEN 1770
1740 FOR I=59 TO 60
1750 CALL PATTERN(#10,I)
1760 NEXT I :: GOTO 1470
1770 FOR I=139 TO 140
1780 CALL PATTERN(#10,I)
1790 NEXT I
1800 GOTO 1470
1810 !----NUAGES-----------
1820 FOR A=2 TO 9
1830 IF RA<>0 THEN 1860
1840 DISPLAY AT(6,1)SIZE(17)BEEP:"evites
les nuages"
1850 DISPLAY AT(8,1)SIZE(14):"et la monta
gne"
1860 CALL SPRITE(#A,92,16,240,(RND*210)+4
0,-8,0)
1870 FOR I=0 TO 60 :: NEXT I
1880 CALL COINC(ALL,C):: IF C=-1 THEN CAL
L DELSPRITE(#A):: GOTO 1860
1890 NEXT A
1900 RETURN
1910 !---EXPLOSION---------
1920 CALL MOTION(#10,0,0)
1930 CALL SOUND(600,-8,0,110,12,110,12,80
0,30)
1940 CALL COLOR(#10,16)
1950 CALL POSITION(#10,X,Y)
1960 CALL SPRITE(#1,44,10,X,Y)
1970 FOR I=1 TO 300 :: NEXT I
1980 CALL LOCATE(#10,160,26):: CALL DELSP
RITE(#1)
1990 CA=CA+1 :: CALL SOUND(200,120,0):: C
ALL HCHAR(24,10-CA,32)
2000 IF IPO=1 THEN AA=AA+1 :: CALL SOUND(
300,150,2):: CALL HCHAR(24,24+AA,127)::
 IPO=0 :: IF AA=3 AND RA<>0 THEN 2820
2010 IF AA=3 THEN 2030
2020 IF CA<>5 THEN 1470
2030 FOR I=1 TO 200 :: NEXT I
2040 !--PLUS D'HELICOPTERE--
2050 DISPLAY AT(10,1)BEEP ERASE ALL:"PREN
EZ DES COURS DE PILOTAGE"
2060 DISPLAY AT(16,1):"   ENCORE UN ESSAI
 OxN"
2070 CALL KEY(0,K,S):: IF S=0 THEN 2070
2080 IF K=79 OR K=111 THEN 2090 ELSE CALL
 CLEAR :: END
2090 TT=0 :: IPO=0 :: RA=0 :: AA=0 :: CA=
0 :: CALL CLEAR :: CALL DELSPRITE(ALL):
: GOTO 650
2100 !---TABLEAU 2---------
2110 CALL SOUND(500,262,2,145,5,392,3)
2120 CALL SOUND(50,262,30)
2130 FOR I=1 TO 3 :: CALL HCHAR(5+I,30,12
6):: CALL SOUND(-80,400*I,11):: NEXT I
2140 GOSUB 2900
2150 CALL SPRITE(#11,128,15,145,229)
2160 IF AA>=1 THEN DISPLAY AT(21,18):"au
suivant" :: GOTO 2180
2170 DISPLAY AT(21,20):"grimpe"
2180 CALL KEY(0,K1,S)
2190 CALL COINC(ALL,C):: IF C=-1 THEN 238
0
2200 CALL POSITION(#11,X,Y)
2210 IF Y>160 AND Y<216 THEN IF Y-X>52 AN
D Y-X<60 THEN 2380
2220 IF Y-X>196 AND Y-X<204 THEN 2380
2230 IF Y>166 AND Y<250 THEN IF Y+X*5/7>2
45 AND Y+X*5/7<251 THEN 2380
2240 IF Y>250 OR X>175 THEN 2380
2250 CALL COINC(ALL,C):: IF C=-1 THEN 238
0
2260 IF K1=68 OR K1=69 OR K1=83 THEN 2300
2270 CALL MOTION(#11,0,0)
2280 FOR I=128 TO 135 :: CALL PATTERN(#11
,I):: NEXT I
2290 GOTO 2180
2300 IF K1=69 THEN CALL MOTION(#11,-8,0):
: GOTO 2340
2310 IF K1=68 THEN CALL MOTION(#11,0,4)::
 GOTO 2340
2320 IF K1=83 THEN CALL MOTION(#11,0,-4):
: GOTO 2340
2330 IF K1=88 THEN CALL MOTION(#11,8,0)
2340 FOR I=36 TO 43 :: CALL PATTERN(#11,I
):: NEXT I
2350 CALL POSITION(#11,X,Y):: IF Y>228 AN
D Y<236 THEN IF X<64 THEN 2530
2360 GOTO 2180
2370 !---CHUTE--------------
2380 CALL MOTION(#11,16,0)
2390 FOR I=1 TO 20 :: CALL SOUND(-200,-8,
0,222,30,222,30,200*I,30)
2400 CALL POSITION(#11,X,Y)
2410 IF X>200 THEN CALL DELSPRITE(#11)::
GOTO 2430
2420 NEXT I
2430 AA=AA+1 :: CALL SOUND(300,150,2):: C
ALL HCHAR(24,25+AA,127)
2440 IF AA<>3 THEN 2140
2450 IF RA<>0 THEN 2820
2460 DISPLAY AT(21,18)
2470 FOR I=1 TO 200 :: NEXT I
2480 DISPLAY AT(10,1)BEEP ERASE ALL:"VOUS
 MANQUEZ D'EXERCICE": : : : :" ENCORE
UN ESSAI  OxN"
2490 CALL KEY(0,K,S):: IF S=0 THEN 2490
2500 IF K=79 OR K=111 THEN 2510 ELSE END
2510 TT=0 :: RA=0 :: IPO=0 :: CA=0 :: AA=
0 :: CALL CLEAR :: CALL DELSPRITE(ALL):
: GOTO 650
2520 ! --ASCENSION REUSSIE--
2530 CALL LOCATE(#11,48,228)
2540 CALL SOUND(100,2000,2)
2550 CALL SOUND(100,3000,2)
2560 CALL LOCATE(#11,32,228)
2570 CALL SOUND(200,4000,2)
2580 CALL DELSPRITE(#11)
2590 CALL SPRITE(#10,140,16,17,225)
2600 CALL HCHAR(3,29,32,2)
2610 CALL HCHAR(4,29,32,2)
2620 FOR I=1 TO 3 :: CALL HCHAR(5+I,30,32
):: CALL SOUND(-80,400*I,11):: NEXT I
2630 DISPLAY AT(21,18)
2640 IPO=1 :: GOTO 1470
2650 !--MUSIQUE DE REUSSITE-
2660 RESTORE 200
2670 FOR I=1 TO 10
2680 READ F1,D1
2690 F2=F1*3
2700 F3=F1/2
2710 D1=D1*85
2720 CALL SOUND(D1,F1,2,F2,5,F3,5)
2730 NEXT I
2740 RA=RA+1 :: CALL HCHAR(22,6+RA,124)::
 AA=AA+1
2750 IPO=0
2760 IF RA=3 OR AA=3 THEN 2810
2770 FOR A=2 TO 9 :: CALL DELSPRITE(#A)::
 NEXT A :: V=V+3 :: TT=TT+2 :: GOSUB 18
20
2780 DISPLAY AT(6,1)SIZE(17)
2790 DISPLAY AT(8,1)SIZE(14)
2800 GOTO 1470
2810 !----FIN DE JEU--------
2820 FOR I=1 TO 200 :: NEXT I
2830 IF RA=3 AND CA<=2 THEN 2870
2840 DISPLAY AT(5,1)BEEP ERASE ALL:"    M
ISSION EFFECTUEE    ": :"CRASH DE:";CA
;" HELICOPTERE S ": :" MORT DE:";AA-RA;"
ALPINISTE S "
2850 DISPLAY AT(11,1):"  VOUS POUVEZ MIEU
X FAIRE  "
2860 GOTO 2060
2870 DISPLAY AT(7,1)BEEP ERASE ALL:"== VO
US ETES UN CHAMPION ==": :"   VOUS N'AV
EZ CASSE QUE": :"    ";CA;"HELICOPTERE S"
2880 GOTO 2060
2890 !---ATTENTE---------
2900 DISPLAY AT(3,1)SIZE(25):"appuies sur
 ""Q"" et grimpe"
2910 DISPLAY AT(5,1)SIZE(21):"en evitant
 les nuages"
2920 CALL KEY(1,K,S):: IF K<>18 THEN 2920
2930 DISPLAY AT(3,1)SIZE(25)
2940 DISPLAY AT(5,1)SIZE(21)
2950 RETURN
2960 !---ECRAN TITRE--------
2970 FOR I=2 TO 11 :: CALL COLOR(I,2,2)::
 NEXT I
2980 CALL MAGNIFY(2)
2990 DISPLAY AT(1,1)ERASE ALL:"SOS HELICO
"
3000 FOR I=1 TO 10
3010 CALL GCHAR(1,2+I,CAR)
3020 XV=SGN(RND-.5)*INT(RND*20)+9 :: YV=S
GN(RND-.5)*INT(RND*20)+9
3030 CALL SPRITE(#I,CAR,2+I,80,110,XV,YV)
3040 RANDOMIZE :: CALL SOUND(80,110*I,5)
3050 NEXT I
3060 CALL DELSPRITE(#4)
3070 FOR I=1 TO 100 :: NEXT I
3080 FOR I=1 TO 3
3090 CALL SOUND(100,1500-100*I,5)
3100 CALL MOTION(#I,0,0)
3110 CALL LOCATE(#I,80-A,78+16*I):: IF A=
8 THEN A=0 :: GOTO 3130
3120 A=8
3130 NEXT I
3140 A=0
3150 FOR I=5 TO 7
3160 CALL SOUND(100,1500-100*I,5)
3170 CALL MOTION(#I,0,0)
3180 CALL LOCATE(#I,112+A,70+2*B)
3190 A=A+10 :: B=B+8 :: NEXT I
3200 FOR I=8 TO 10
3210 CALL SOUND(100,1500-100*I,5)
3220 CALL MOTION(#I,0,0)
3230 CALL LOCATE(#I,132-C,70+2*B)
3240 C=C+10 :: B=B+8 :: NEXT I
3250 A,B,C=0
3260 FOR A=1 TO 10
3270 CALL COLOR(#A,1)
3280 CALL SOUND(100,400,10)
3290 CALL COLOR(#A,A+3)
3300 NEXT A
3310 CALL CLEAR :: FOR I=1 TO 300 :: NEXT
 I
3320 FOR I=2 TO 11 :: CALL COLOR(I,16,2):
: NEXT I :: FOR I=1 TO 10 :: CALL COLOR
(#I,1):: NEXT I
3330 DISPLAY AT(4,1):"LE JEU CONSISTE A R
ECUPERER": :"TROIS ALPINISTES BLOQUES P
AR": :"LE MAUVAIS TEMPS EN ALTITUDE": :"L
E PILOTAGE N'EST PAS FACILE"
3340 DISPLAY AT(12,1):"CAR L'HELICOPTERE
POSSEDE": :"UNE CERTAINE INERTIE DE PLU
S": :"LA MONTAGNE EST PLEINE DE": :"PIEGE
S.UTILISEZ LE CLAVIER"
3350 DISPLAY AT(23,1):"<ENTER>"
3360 CALL KEY(0,K,S):: IF S=0 THEN 3360
3370 CALL CLEAR
3380 FOR I=1 TO 10 :: CALL COLOR(#I,I+2):
: NEXT I
3390 FOR I=1 TO 100 :: NEXT I :: RETURN
```

Le mode édition en Basic TI

Normalement, pour passer en mode "édition", il faut utiliser la commande "EDIT numéro de ligne". Ceci peut être avantageusement remplacé par :

Numéro de ligne et FCTN-E ou FCTN-X (touche fléchées).

Lorsque vous êtes en mode édition, vous pouver corriger la ligne ou passer à une autre ligne en employant les commandes précitées. Pour revenir au mode commande, il suffit d'appuyer sur la touche "ENTER".

Utilisation de la Mini-mémoire

Denise Amrouche

En dehors des utilisations les plus évidentes (sauvegarde rapide de courts programmes en Basic et écriture de programmes en assembleur), la Mini-mémoire apporte d'autres possibilités :

1 - Création de fichiers d'autres types que ceux possibles sur cassettes. Nous montrerons essentiellement comment ces fichiers permettent d'augmenter la taille mémoire du TI-99/4A, des 4Ko de la Mini-mémoire.

2 - Accès à la mémoire centrale de l'ordinateur. Nous donnerons quelques exemples.

3 - Accès à la mémoire VDP (mémoire vive du processeur video) qui gère ce qui concerne l'écran. Nous détaillerons les possibilités ainsi ouvertes, pour accélérer la gestion de l'écran en Basic, et pour créer quelques lutins (sprites).

Fichiers en Mini-mémoire

Rappelons brièvement les différentes caractéristiques d'un fichier, leurs avantages et leurs inconvénients (pour plus de détails, voir le manuel d'utilisation du TI-99/4A, pages 122 et suivantes).

SEQUENTIAL ou RELATIVE

Un fichier ouvert en mode SEQUENTIAL sera lu dans l'ordre successif des différentes fiches. Ce sera le cas de notre premier exemple, où nous avons besoin de lire le fichier une seule fois, d'un bout à l'autre. C'est le seul choix possible sur cassette, et cela impose une lecture assez longue si l'on désire lire seulement la fiche 30 (par exemple), mais cela aura l'avantage de permettre un fichier de type VARIABLE en Mini-mémoire, et de gagner ainsi de la place.

A l'opposé, un fichier ouvert en mode RELATIVE permet, grâce à la clause REC, l'accès direct à une fiche de numéro donné. Ce sera le cas de notre second exemple. Un tel fichier ne pourra être ouvert que FIXED (longueur fixe). Il faudra donc, dans la mesure du possible, fixer cette longueur de façon à gagner de la place mémoire.

INPUT, OUTPUT, UPDATE et APPEND

En Mini-mémoire, nous utiliserons UPDATE, qui permet indifféremment d'écrire ou de lire dans le fichier.

DISPLAY ou INTERNAL

En type DISPLAY, tout se passe comme sur l'écran. C'est-à-dire, par exemple, que 12 est stocké comme la suite des codes ASCII 32, 49, 50 et 32, tout nombre sur l'écran étant précédé et suivi d'un espace. Si on demande "PRINT #1:12,34", le fichier notera 1 espace, 1, 2, des espaces jusqu'à la moitié d'une ligne d'écran, 1 espace, 3, 4, et 1 espace, ce qui prendra énormément de place. Ceci explique l'erreur commise page 15 du mode d'emploi jaune de la Mini-mémoire. A la ligne "PRINT #3", il faut mettre "PRINT #3:A:B:C:D", pour séparer les quatre enregistrements.

Au contraire, en type INTERNAL, les caractères sont codés comme dans la machine. Une chaîne A$ occupe LEN(A$)+1 octets (l'octet supplémentaire servant à noter la longueur de la chaîne).

Un nombre A occupe toujours 9 octets : 1 pour sa longueur et 8 pour son codage en virgule flottante. On remarque donc qu'un nombre, même entier ou petit, prend toujours 9 octets alors que le nombre 123 mis en chaîne ne prendra plus que 4 octets. Cette remarque sera utilisée dans l'exemple 1.

VARIABLE ou FIXED

Un fichier SEQUENTIAL, pourra être en VARIABLE.

Pour un fichier RELATIVE, FIXED est imposé et, pour les raisons déjà citées, il sera bon de préciser cette longueur fixe.

Exemple 1 : Dessin d'une voiture de course

En Basic simple, la Mini-mémoire ne peut pas être utilisée pour ajouter des lignes de Basic à un programme qui remplirait déjà la console, mais vous pouvez utiliser la Mini-mémoire en ouvrant un fichier pour y stocker des DATA qui prennent beaucoup de place dans le programme.

L'exemple 1 est un programme qui doit dessiner sur l'écran une belle voiture de course, qui pourra vous servir d'introduction à un programme de course automobile. Il y a beaucoup de caractères à redéfinir (CALL CHAR), et ensuite beaucoup de caractères à mettre en place (CALL HCHAR).

Notre démarche sera la suivante :

1 - insérer la Mini-mémoire (sauvegarder son contenu, s'il y a lieu);

2 - taper le programme FICHFOR, dans lequel se trouvent tous les DATA qui seront lus et transférés sur un fichier en Mini-mémoire;

3 - faire RUN, ce qui effectue le transfert;

4 - sauvegarder le programme sur cassette ou disquette;

5 - taper le programme FORMULE1, qui lira le fichier et fera le dessin sur l'écran;

6 - faire RUN;

7 - sauvegarder le programme sur disquette ou cassette.

Si le dessin est joli et s'il n'y a pas d'erreur, sauver le fichier Mini-mémoire sur cassette en utilisant EASY BUG S7000 TO? 8000.

S'il y a une erreur de graphisme, bien vérifier le programme FORMULE1. Si cela ne va toujours pas, recharger FICHFOR, le vérifier et recommencer la procédure en 2.

Pour une utilisation ultérieure, il vous suffira de charger le fichier en Mini-mémoire (EASY BUG option L), puis le programme FORMULE1 en Basic. Vous avez un programme Basic très court, pour un dessin assez compliqué, et il vous reste de la place pour faire un beau programme.

Commentaires sur le programme FICHFOR

```
100 REM *****************
105 REM *  Programme    *
110 REM *   Fichfor     *
115 REM *****************
120 REM
125 REM Data pour les
130 REM CALL CHAR
135 REM
180 DATA 0000000000000007,000000
00000000FF,00000000001011E1,00000
000F8FC0C02
```



```
190 DATA 000000000000007F,000000
00000000F,0000000000000F3F,38271
0100808FEFF
200 DATA 00807F0000007F8,0001FE0
00000FF,41810000000FF,FF0183770
E0CFC0C
210 DATA FFFFFF1F0F6F7773,FCFFFF
FFFFFFFFFF,00F00F80FFC0FFC,00000
0E01E01FC03
220 DATA 0000000000C03EC1,000000
01030707FF,007FFFFFFFC79B3D,00FC
FEFFFFFFFFFF
230 DATA 000000080C0C0E0,3F3F67
C7CB9B9D9D,FFFFFFFFFFFFFFFF,FFE0
E0FFF0FFF9F2
240 DATA FF0F00008040C02,F807FE7
C7C7C7C7C,1C18989898988C,7B7B7
7F7F7F7FB7B
250 DATA E0E0E7E4E4E7E0E,3F1F800
000FF,C0807F000000FF,000000F8070
0FF
260 DATA E51F0000807BFF,FF03F807
,E0E038867907,000000000806018
270 DATA 9D9D9D898BC3476F,F4F5F5
F5F5FFF0C,E0202040BFFF,7C7C7C7CF
FFF
280 DATA 8C8E8E8707FF3F0F,370F0F
FFFFFFFFFF,E0C0C0FFF880801F,0000
00FF000000FF
290 DATA 0000FF00000000FF,0000FF
07030100FF,000FFFFFFFFFFFC,07FFF
FFFFFFEFC
300 DATA FFE0E0C08,E,3F3F1F,FFFF
FF,C0BF00FF,00FF008,00FF,0FF,FF3
F,FE8
310 REM
315 REM Data pour
320 REM positionner les
325 REM caracteres sur
330 REM l'ecran
335 REM
340 DATA 1210102,1211103,1212103
,1213104,1214105,1215106,1216107
350 DATA 1309108,1310109,1311110
,1312111,1313112,1314113,1315114
,1316115
360 DATA 1317116,1318117,1319118
,1320119,1321120,1322121,1323122
370 DATA 1409123,1410124,1411125
,1412126,1413127,1414128,1415129
,1416124
380 DATA 1417130,1418131,1419132
,1420133,1421134,1422135,1423136
,1424137
390 DATA 1509138,1510124,1511139
,1512140,1513141,1514142,1515143
,1516124
400 DATA 1517144,1518145,1519146
,1520147,1521148,1522149,1523150
,1524151
410 DATA 1609152,1610153,1611154
,1612155,1613156,1614157,1615158
,1616159
415 REM
420 REM Lecture des Data
425 REM et mise en
430 REM fichier
435 REM
450 CALL INIT
460 OPEN #1:"MINIMEM",SEQUENTIAL
,VARIABLE,UPDATE,INTERNAL
470 FOR I=102 TO 159
480 READ A$
490 PRINT #1:A$
500 NEXT I
510 FOR I=1 TO 62
520 READ B
530 PRINT #1:B
540 NEXT I
550 CLOSE #1
560 STOP
```

Le programme utilise les caractères 102 à 159; les lignes 180 à 300 donnent les patrons de chacun de ces caractères.
Pour placer ces caractères, nous aurons besoin de la ligne C et du No CH du caractère. Pour éviter de stocker trois nombres par caractère, ce qui prendrait 27 octets, nous utiliserons un seul nombre, codé de cette façon :

- ligne : 2 chiffres;
- colonne : 2 chiffres;
- caractère : 3 chiffres.

Ainsi, 1210102 correspond à la ligne 12, la colonne 10, et le caractère 102.

Les lignes 450 à 550, après avoir réinitialisé la Mini-mémoire, lisent les DATA et les mettent dans un fichier.

Commentaires sur le programme FORMULE1

```
100 REM ***************
105 REM *  Programme   *
110 REM *   Formule1   *
115 REM ***************
120 REM
125 REM Dessin en
130 REM utilisant le
135 REM fichier de
140 REM donnees
145 REM
160 CALL CLEAR
170 OPEN #1:"MINIMEM",SEQUENTIAL
,UPDATE,VARIABLE,INTERNAL
180 FOR I=102 TO 159
190 INPUT #1:A$
200 CALL CHAR(I,A$)
210 NEXT I
220 FOR I=1 TO 62
230 INPUT #1:B
240 L=INT(B/100000)
250 U=B-L*100000
260 C=INT(U/1000)
270 CH=U-C*1000
280 CALL HCHAR(L,C,CH)
290 NEXT I
300 CLOSE #1
310 GOTO 310
```

La démarche est inverse, le fichier est lu et utilisé d'abord pour définir les caractères (180 à 210), puis pour les placer (220 à 290). On doit alors décoder notre nombre de sept chiffres, pour en extraire L, C et CH. On a évité les fonctions STR$ et SEG$ pour des raisons de rapidité d'exécution.

Exemple 2 : Airs de chansons enfantines

La musique est un autre exemple utilisant beaucoup de DATA, donc beaucoup de place. Dans le même esprit que dans le premier exemple, la procédure sera la suivante :

1 - insérer la Mini-mémoire (sauvegarder son contenu s'il y a lieu);
2 - taper le programme FICHMUS;
3 - faire RUN;
4 - le sauvegarder sur cassette ou disquette;
5 - taper le programme musique et le tester.

Si le résultat est bon, sauvegarder le programme MUSIQUE, puis sauvegarder le fichier Mini-mémoire (EASY BUG S 7000 TO? 8000).
Si le résultat n'est pas au point, vérifier MUSIQUE, le sauvegarder sur cassette, puis recharger FICHMUS et le vérifier. Refaire RUN, sauvegarder FICHMUS, charger MUSIQUE. Tout devrait être bon. Pour une utilisation ultérieure : charger la Mini-mémoire (EASY BUG option L), puis le programme MUSIQUE; ensuite, faire RUN.

Commentaires sur le programme FICHMUS

```
100 REM ***************
105 REM *  Programme   *
110 REM *   Fichmus    *
115 REM ***************
120 REM
125 REM Donnees pour
130 REM "Au clair de
135 REM la lune"
140 REM
145 REM Fiches 0 a 11
150 REM
170 DATA 1,392,196,196,1,392,196
,196,1,392,294,247,1,440,294,247
```

```
,2,494,196,196,2
,440,294,262,1,392,196,196
180 DATA 1,494,196,196,1,440,294
,262,1,440,294,262,3,392,196,196
,1,40000,40000,4
0000
185 REM
190 REM Fiches 12 a 35
195 REM
200 DATA 1,440,139,139,1,440,139
,139,1,440,220,196,1,440,220,196
,2,330,139,139,2
,330,220,196
210 DATA 1,440,139,139,1,392,139
,139,1,370,220,196,1,330,220,196
,3,294,147,147,1
,40000,40000,40000
220 DATA 1,392,196,196,1,392,196
,196,1,392,294,247,1,440,294,247
,2,494,196,196,2
,440,294,262
230 DATA 1,392,196,196,1,494,196
,196,1,440,294,262,1,440,294,262
,3,392,294,247,1
,40000,40000,40000
235 REM
240 REM "Meunier, tu
242 REM dors"
245 REM
250 REM Fiches 36 a 77
255 REM
280 DATA 2,440,40000,40000,4,440
,175,175,2,523,262,220,4,349,131
,131,1,330,262,2
20,1,349,262,220
290 DATA 4,392,165,165,1,349,262
,233,1,392,262,233,2,440,175,175
,2,349,262,220,2
,440,262,220
300 DATA 4,440,175,175,2,523,262
,220,4,349,131,131,1,330,262,220
,1,349,262,220,4
,392,165,165,1,349,262,233
310 DATA 1,392,262,233,4,349,262
,220,6,40000,40000,40000,1,440,4
0000,40000
320 DATA 1,440,40000,40000,2,440
,175,40000,1,440,262,220,1,440,2
62,220,2,440,131
,40000,1,440,262,220
330 DATA 1,440,262,220,4,523,175
,40000,2,440,131,40000,1,440,262
,220,1,440,262,2
20,2,392,165,40000
340 DATA 1,392,262,233,1,392,262
,233,2,523,131,40000,1,523,262,2
33,1,523,262,233
,4,440,175,40000
350 DATA 2,40000,131,40000
355 REM
360 REM "Frere Jacques"
365 REM
370 REM Fiches 78 a 81
375 REM
400 DATA 2,392,247,247,2,440,262
,262,2,494,294,294,2,392,247,247
405 REM
410 REM Fiches 82 a 84
415 REM
420 DATA 2,494,196,196,2,523,220
,220,4,587,247,196
425 REM
430 REM Fiches 85 a 90
435 REM
440 DATA 1,587,247,247,1,659,247
,247,1,587,262,262,1,523,262,262
,2,494,294,294,2
,392,247,247
445 REM
450 REM Fiches 91 a 93
455 REM
460 DATA 2,392,247,247,2,294,196
,196,4,392,247,247
470 REM
480 REM Mise sur fichier
490 REM
500 CALL INIT
510 OPEN #1:"MINIMEM",RELATIVE,F
IXED 36,INTERNAL,UPDATE
520 FOR I=1 TO 94
530 READ A,B,C,D
540 PRINT #1:A,B,C,D
550 NEXT I
560 CLOSE #1
570 STOP
```

Nous ouvrirons cette fois un fichier RELATIVE, donc avec possibilité d'accès direct à une fiche donnée, d'une part parce que trois airs sont proposés, d'autre part parce qu'un même air peut reprendre plusieurs fois la même ligne de partition.
Chaque fiche (ligne 540) enregistrera les renseignements concernant une note :
- coefficient de durée;
- note;
- première note d'accompagnement;
- deuxième note d'accompagnement.

Chaque fiche aura donc une longueur de 36 octets (9*4).

Pour garder une vitesse d'exécution de la musique à peu près satisfaisante, on a renoncé à tout codage ou traduction en chaîne des notes. Il n'était pas question non plus de lire toutes les notes avant exécution, ce qui aurait lassé l'auditeur impatient, et occupé de la place en mémoire, alors que notre but est d'en économiser dans le programme principal.

Commentaires sur le programme MUSIQUE

```
100 REM ***************
105 REM *  Programme   *
110 REM *   Musique    *
115 REM ***************
120 REM
125 REM Musique a partir
130 REM d'un fichier en
135 REM Mini-memoire
140 REM
150 OPEN #1:"MINIMEM",RELATIVE,F
IXED 36,INTERNAL,UPDATE
160 CALL CLEAR
170 PRINT : : : :"CHOISISSEZ:":
:
180 PRINT "1  AU CLAIR DE LA LUN
E": :
190 PRINT "2  MEUNIER,TU DORS":
:
200 PRINT "3  FRERE JACQUES": :
:
210 CALL KEY(0,K,S)
220 IF (K<1)+(K<49)+(K>51)<>0 TH
EN 210
230 ON K-48 GOTO 260,400,510
240 REM
245 REM Au clair de la
250 REM lune
255 REM
260 L=250
270 FOR J=1 TO 2
280 FOR I=0 TO 11
290 GOSUB 780
300 NEXT I
310 NEXT J
320 FOR I=12 TO 35
330 GOSUB 780
340 NEXT I
350 GOTO 210
360 REM
370 REM Meunier, tu dors
380 REM
400 L=200
410 FOR I=36 TO 77
420 GOSUB 780
430 NEXT I
440 FOR I=58 TO 77
450 GOSUB 780
460 NEXT I
470 GOTO 210
480 REM
490 REM Frere Jacques
500 REM
510 L=300
520 FOR K=1 TO 2
530 FOR J=1 TO 2
540 FOR I=78 TO 81
550 GOSUB 780
```

```
560 NEXT I
570 NEXT J
580 FOR J=1 TO 2
590 FOR I=82 TO 84
600 GOSUB 780
610 NEXT I
620 NEXT J
630 FOR J=1 TO 2
640 FOR I=85 TO 90
650 GOSUB 780
660 NEXT I
670 NEXT J
680 FOR J=1 TO 2
690 FOR I=91 TO 93
700 GOSUB 780
710 NEXT I
720 NEXT J
730 NEXT K
740 GOTO 210
750 REM
755 REM Sous-programme
760 REM de lecture et de
765 REM jeu des notes
770 REM
780 INPUT #1,REC I:A,B,C,D
790 CALL SOUND(L*A,B,1,C,7,D,10)
800 RETURN
```

Pour chaque air, on utilise la clause REC, avec la variable I qui indique la fiche que l'on désire lire. On peut ainsi relire des fiches... Le programme comporte de nombreuses lignes courtes et occupe peu de place en mémoire.

Accès à la mémoire centrale

La Mini-mémoire étant insérée, vous disposez des deux routines : CALL PEEK et CALL LOAD.

CALL PEEK(adresse décimale, variable X, variable Y, ...) lit le contenu de l'octet situé à l'adresse indiquée, et le met en valeur décimale dans X, lit le contenu de l'adresse suivante et le met dans Y, ... Plusieurs adresses peuvent être lues, séparées par "".

CALL LOAD(adresse décimale, X, Y, ...) a l'effet inverse. Il met dans l'adresse indiquée l'octet correspondant à la valeur décimale X, et dans l'adresse suivante l'octet correspondant à la valeur décimale Y, ...

Quelques adresses intéressantes

A) −31888 et −31887.

Dans ces deux adresses consécutives (>8370 et >8371 en hexa) se trouve la dernière adresse VDP utilisable.
Faites en Basic CALL PEEK(−31888,X,Y) puis PRINT X,Y.

1) Si vous n'avez pas de contrôleur de disquettes, vous lisez X=63, Y=255. C'est-à-dire que dans l'adresse −31888 (>8370) se trouve la valeur >3F, et dans −31887 (>8371) la valeur >FF.
La plus haute adresse utilisable en mémoire vive du processeur vidéo est donc >3FFF ou, en décimal, 63*256+255=16383, ce qui est effectivement la plus haute adresse de la MEV VDP (cf. mode d'emploi jaune de la Mini-mémoire, pages 75 et 76).

2) Si vous avez un contrôleur de disquettes, vous obtenez X=55 et Y=215. La plus haute adresse disponible dans la MEV VDP est donc 55*216+215=14295 ou >37D7.
Si, avec un contrôleur de disquettes, vous tapez dans l'ordre :
CALL FILES(1)
NEW
CALL PEEK(−31888,X,Y)
PRINT X,Y
vous obtenez X=59 et Y=227. La plus haute adresse utilisable est donc 59*256+227=15331.

N.B. En Basic simple, le programme est stocké en MEV VDP, il est donc intéressant d'y avoir un maximum de place disponible, d'où l'intérêt si vous possédez un lecteur de disquettes, de taper CALL FILES(1) et NEW avant de taper ou de charger un long programme Basic, ou un programme utilisant de grands ou de nombreux tableaux. Vous récupérez ainsi 15331−14295=1036 octets. Il vous manque encore 16383−15331=1052 octets, par rapport à ceux qui utilisent les cassettes.

B) −31878 (>837A).

C'est dans cette adresse que l'on doit indiquer à l'ordinateur le nombre de lutins (sprites) en mouvement.
Si nous mettons en mouvement trois lutins, nous ferons :
CALL LOAD(−31878,3)

C) −31788 (>83D4).

Une copie du contenu de cette adresse est mise dans le registre 1 du processeur vidéo, chaque fois qu'une touche est pressée alors que l'ordinateur est en mode de scrutation, c'est-à-dire qu'il attend la frappe d'une touche. Or, le registre 1 est celui qui détient des renseignements sur le mode d'affichage et sur la taille des lutins. Nous pouvons donc, en utilisant l'adresse −31788, agir sur ces facteurs.

En détail, en numérotant les bits de 0 à 7, et de gauche à droite :

Bit 0 : 1 toujours
Bit 1 : 1 toujours
Bit 2 : 1 en Basic
Bit 3 : 1 = affichage en mode texte, 0 = autre affichage
Bit 4 : 1 = affichage en mode multicolore, 0 = autre affichage
Bit 5 : 0 toujours
Bit 6 : 0 = lutins définis par un seul caractère, 1 = lutins définis par quatre caractères
Bit 7 : 0 = lutins de dimension normale, 1 = lutins agrandis

Montrons un peu ce qu'est le mode multicolore.

- Remplissez votre écran en tapant n'importe quoi.
- Tapez CALL LOAD(−31788,232).
- Faites ENTER.
- Appuyez sur n'importe quelle touche.
 L'écran est à présent coloré, chaque caractère s'étant divisé en quatre carrés de couleur. C'est le mode multicolore.
 232=>E8=11101000 (le bit 4 est à 1).

Pour revenir à l'état normal :

- Tapez en aveugle CALL LOAD(−31788,224) (224=>E0=11100000), ENTER puis appuyez sur une touche;
- Ou éteignez.

Par défaut, le mode texte est transparent sur fond cyan; il n'est donc pas possible, sans un petit programme en assembleur, de visualiser ce mode d'affichage.
Nous vous avons présenté les adresses les plus utiles en Basic. Il y en a bien sûr d'autres ayant un usage bien particulier pour l'écriture de programmes en assembleur.

Accès à la mémoire vive du processeur vidéo

La mémoire du processeur vidéo est celle qui gère l'écran. En plus, elle contient le programme. En Basic Etendu, si vous disposez d'une extension de mémoire, une partie du programme y est stocké. Vous disposez, la Mini-mémoire étant insérée, de deux instructions d'accès à la MEV VDP.

CALL PEEKV(adresse décimale,X,Y,...) lit à l'adresse indiquée et met le premier octet lu dans X en base 10, le second dans Y en base 10... Des adresses non consécutives peuvent être explorées en séparant par "".
CALL POKEV(adresse décimale,X,Y,...) a l'effet inverse. Il met à partir de l'adresse indiquée les valeurs décimales X, Y,...

Nous allons donner de nombreux exemples d'utilisation. Signalons que les adresses données sont celles utilisées par le Basic (en assembleur, la disposition des différentes tables est autre, et dépend du mode graphique

utilisé); en outre, dans cette MEV VDP, les codes ASCII sont décalés de 96. C'est-à-dire que, par exemple, l'espace (code ASCII 32) est codé 32+96 soit 128. Nous dirons que 128 est le code ASCII interne pour le caractère "espace".

Table d'écran

Cette table, qui s'étend des adresses 0 à 767 (>0000 à >02FF), accorde une adresse à chaque emplacement de l'écran, dans laquelle elle met le code ASCII interne du caractère occupant cet emplacement. Les adresses 0 à 31 correspondent aux 32 colonnes de la première ligne, les adresses 32 à 63 aux 32 colonnes de la seconde ligne, etc...

Exemple : votre écran étant vide, tapez :
CALL PEEKV(4,X)
PRINT X
Vous obtenez X=128 car, à l'emplacement d'adresse 4 (donc ligne 1 - colonne 5), il y a un espace (code ASCII interne 128).

Autre exemple : tapez CALL POKEV(47,161)
Vous obtenez un A sur la ligne 2 - colonne 16, car le code ASCII interne de A est 65+96 soit 161.

Une règle : pour afficher le caractère de code ASCII H à la ligne L (1<=L<=24), colonne C (1<=C<=32), il faut mettre H+96 à l'adresse 32*L+C−33, ce qui se fait par CALL POKEV(32*L+C−33,H+96).

L'instruction habituelle CALL HCHAR(L,C,H,2) s'écrira CALL PEEKV(32*L+C−33,H+96,H+96).

L'instruction CALL GCHAR(L,C,X) s'écrira CALL PEEKV(32*L+C−33,X), où X est égal à X−96.

Cette traduction présente dans certains cas deux avantages :

1 - gain de place en mémoire;
Exemple : pour dessiner un échiquier ou un damier, vous avez à mettre en place un dessin sur un carré de deux caractères par deux. Programme habituel :
CALL HCHAR(X,Y,A)
CALL HCHAR(X,Y+1,B)
CALL HCHAR(X+1,Y,C)
CALL HCHAR(X+1,Y+1,D)

Ceci peut être remplacé par :
CALL POKEV(32*X+Y−33,A+96,B+96,"",32*X+Y−1,C+96,D+96)

2 - rapidité d'exécution

```
100 REM ****************
101 REM *Affichage     *
102 REM *d'une chaine  *
103 REM *en un endroit *
104 REM *donne         *
105 REM ****************
106 REM
110 CALL CLEAR
120 A$="1004BONJOUR,COMMENT ALLE
Z-VOUS?"
130 GOSUB 5000
200 GOTO 200
5000 REM
5002 REM Sous-programme
5004 REM d'affichage
5006 REM
5010 L=VAL(SEG$(A$,1,2))
5020 C=VAL(SEG$(A$,3,2))
5030 LO=LEN(A$)-5
5040 FOR I=0 TO LO
5050 CALL HCHAR(L,C+I,ASC(SEG$(A
$,5+I,1)))
5060 NEXT I
5070 RETURN
```

Dans ce programme Basic simple et classique, on utilise un sous-programme qui lit la ligne d'affichage, la colonne et le texte. A l'exécution, on voit le message s'afficher peu à peu.
Autre solution :

```
100 REM ****************
101 REM *Affichage      *
102 REM *d'une chaine   *
103 REM *par la         *
104 REM *fonction       *
105 REM *CALL POKEV     *
106 REM ****************
107 REM
110 CALL CLEAR
120 CALL POKEV(291,162,175,174,1
70,175,181,178,129,163,175,173,1
73,165,174,180,1
40,161,172,172,165,186,141,182)
130 CALL POKEV(314,175,181,179,1
59)
200 GOTO 200
```

291=32*10+4−33
162=66+96 (66=code ASCII de B)
Ici, le message s'affiche instantanément.

Table des couleurs

Chaque groupe de caractères se voit attribuer une adresse dans cette table, dans laquelle seront codées la couleur du caractère et la couleur du fond.
En assembleur, les codes des couleurs sont diminués de 1 par rapport à leur code en Basic; ainsi, le transparent devient 0, le noir devient 1, etc...
Pour avoir un groupe de caractères en bleu foncé sur fond vert vert, il faut mettre la valeur >43 dans l'adresse associée à la couleur de ce groupe.

Une règle : pour colorer un groupe donné en caractère de couleur C (codage Basic 1<=C<=16) sur fond de couleur F (1<=F<=16), il faut mettre 16*C+F−17 dans l'adresse associée à ce groupe.

Adresse : 783 (>30F) - gpe 0

Exemple : tapez CALL POKEV(788,11,11,11).
Quand vous faites ENTER, certains caractères de votre texte deviennent provisoirement jaunes, l'effet est temporaire. Exemple : dans un programme, CALL PEEKV à une des adresses indiquées peut vous permettre de connaître la couleur de tel ou tel groupe et d'agir en conséquence. C'est une instruction qui n'existe pas en Basic.

Exemple : vous avez un long texte à afficher sur l'écran et vous ne voulez pas fatiguer le joueur ou lecteur par l'effet de "scroll" (déroulement); vous mettez pour cela tous les caractères en transparent, vous affichez votre texte, puis vous rétablissez les couleurs. Il y a alors un certain effet de papillotement car les couleurs se rétablissent groupe par groupe.

```
100 REM ****************
105 REM *Affichage d'un*
110 REM *texte d'un     *
112 REM *seul coup      *
114 REM ****************
116 REM
120 CALL CLEAR
130 FOR I=1 TO 12
140 CALL COLOR(I,1,1)
150 NEXT I
160 PRINT "ESSAI D'AFFICHAGE D'U
N LONG MESSAGE": : : :
170 PRINT "La quatrieme planete
etait  celle du businessman.Cet
   homme etait s
i occupe qu'il"
180 PRINT "ne leva meme pas la t
ete a  l'arrivee du petit prince
.   _Bonjour,lui
 dit celui-ci."
190 PRINT "Votre cigarette est e
teinte. _Trois et deux font cinq
. Cinq et sept
douze.Douze et"
200 PRINT "trois quinze.Bonjour.
"
250 FOR I=1 TO 12
260 CALL COLOR(I,2,1)
270 NEXT I
300 GOTO 300
```

Remplacez les lignes 130,140 et 150 par l'unique ligne :

130 CALL POKEV(784,0,0,0,0,0,0,0,0,0,0,0,0,0) et les lignes 250, 260 et 270 par l'unique ligne :
290 CALL POKEV(784,16,16,16,16,16,16,16,16,16,16,16,16)

```
100 REM ****************
102 REM *Affichage d'un*
104 REM *texte d'un    *
106 REM *seul coup     *
108 REM ****************
110 REM
120 CALL CLEAR
130 CALL POKEV(784,0,0,0,0,0,0,0,0,0,0,0,0)
160 PRINT "ESSAI D'AFFICHAGE D'UN LONG MESSAGE": : : :
170 PRINT "La quatrieme planete etait  celle du businessman.Cet   homme etait s
i occupe qu'il"
180 PRINT "ne leva meme pas la tete a  l'arrivee du petit prince.  _Bonjour,lui
 dit celui-ci."
190 PRINT "Votre cigarette est eteinte. _Trois et deux font cinq.  Cinq et sept
douze.Douze et"
200 PRINT "trois quinze.Bonjour."
290 CALL POKEV(784,16,16,16,16,16,16,16,16,16,16,16,16)
300 GOTO 300
```

Observez la différence. L'exécution étant plus rapide, il n'y a plus de papillotement.

Table des patrons (descripteurs)

Le caractère ASCII "J" (code ASCII interne J+96) est décrit des adresses 8★(J+96) à 8★(J+96)+7 inclus. Votre patron est une chaîne de 16 caractères alphanumériques (0 à 9, A à F). Vous le coupez en 8 tranches de 2 et, pour utiliser l'instruction CALL POKEV, vous traduisez chaque octet en décimal.
>3A devient 3★16+10=58.

Exemple : le curseur vous gêne ou sa forme ne vous plaît pas, modifiez-le. Le curseur a 30 comme code ASCII; il est donc décrit à partir de l'adresse 8★(30+96)=1008.
Pour le faire disparaître :
CALL POKEV(1008,0,0,0,0,0,0,0,0)
Pour lui donner la forme d'un losange :
CALL POKEV(1008,24,60,126,255,255,126,60,24)
Le patron du losange est "183C7EFFFF7E3C18"
Pour qu'il soit plus gros :
CALL POKEV(1008,255,255,255,255,255,255,255,255)
Cette transformation reste valable jusqu'à ce que vous éteigniez la console ou que vous rétablissiez le curseur habituel par :
CALL POKEV(1008,0,124,124,124,124.124 124,124)

Lutins (sprites)

La gestion des lutins nécessite deux tables :
- la table des attributs, qui garde en mémoire les caractères qui les définissent, leurs couleurs et leurs positions à chaque instant;
- la table des vitesses qui garde en mémoire leurs vitesses horizontales et verticales.

La table des attributs des lutins commence à l'adresse 768 (>300). Chaque lutin occupe quatre adresses consécutives et il faut mettre 208 (>D0) à la fin de cette table pour indiquer qu'il n'y a pas d'autres lutins. Comme la table des couleurs commence à l'adresse 783, si nous voulons simultanément du texte ou du graphisme utilisant des codes ASCII bas, nous ne pouvons utiliser que 3 lutins. Si nous mettons uniquement des lutins, nous pourrons en mettre plus.

Pour chaque lutin, nous devons indiquer successivement la position verticale, la position horizontale, le code ASCII et la couleur.

Position verticale du lutin

255 pour la première ligne (ligne de points), 0 pour la seconde ligne, 1 pour la troisième ligne, et ainsi de suite jusqu'à 190 pour la dernière ligne. Chaque ligne de caractères étant divisée en huit lignes de points, il y a 24★8=192 positions verticales possibles.
Les positions 191 à 254 (sauf 208) sont autorisées, mais le lutin est alors invisible.

Position horizontale du lutin

0 pour la première colonne (colonne de points), 1 pour la deuxième, et ainsi de suite jusqu'à 255 pour la dernière colonne.

Code ASCII

Il s'agit du code interne du caractère dont le patron est utilisé pour la définition du lutin.

Couleur du lutin

Même couleur qu'en Basic moins un (−1).

```
100 REM ****************
101 REM *Exemples de   *
102 REM *sprites en    *
103 REM *Basic simple  *
104 REM *grace a la    *
105 REM *Mini-memoire  *
106 REM ****************
107 REM
110 CALL CLEAR
120 CALL POKEV(768,80,170,180,13)
130 CALL POKEV(772,100,45,169,4)
140 CALL POKEV(776,45,67,153,12)
150 CALL POKEV(780,78,45,153,10,208)
160 CALL LOAD(-31788,224)
170 GOSUB 250
180 CALL LOAD(-31788,225)
190 GOSUB 250
200 CALL LOAD(-31788,226)
210 GOSUB 250
220 CALL LOAD(-31788,227)
230 GOSUB 250
240 GOTO 160
250 CALL KEY(0,K,S)
260 IF S<1 THEN 250
270 RETURN
```

Le programme crée quatre lutins utilisant les caractères prédéfinis par l'ordinateur. En appuyant sur n'importe quelle touche, vous changez la taille des lutins en utilisant ce qui a été dit sur l'adresse −31788.
224=>E0 : petit lutin sur un caractère
225=>E1 : grand lutin sur un caractère
226=>E2 : petit lutin sur quatre caractères
227=>E3 : grand lutin sur quatre caractères
La table de vitesse des lutins commence à l'adresse >780 (1920) et se superpose à la table des patrons. 1920 est le début de la description du caracère ASCII 144, ce qui nous explique qu'en Basic Etendu 143 soit le dernier code ASCII disponible pour définir un caractère. Chaque lutin occupe quatre octets dans la table.

1er octet : vitesse verticale (0 à 255).
De 0 à 127 : vitesse positive.
De −128 à −1 : vitesse négative. Augmentée de 256, donne un entier de 128 à 255.

2ème octet : vitesse horizontale (même convention que pour la vitesse verticale).

3ème et 4ème octets : ces adresses sont utilisées par la machine (routine d'interruption); il est donc inutile de les gérer.

Exemple :

5 CALL CLEAR
10 CALL POKEV(768,80,170,180,13,208)
20 CALL POKEV(1920,100,200)
30 CALL LOAD(−31878,1)
40 GOTO 40
crée un lutin T se déplaçant de haut en bas (vitesse verticale positive) et de droite à gauche (vitesse horizontale négative : 200−256=−56). Après l'arrêt, tapez CALL LOAD(−31878,0) pour faire disparaître tout lutin.

```
100 REM ***************
101 REM *  Exemple de  *
102 REM *  sprites en  *
103 REM * Basic simple *
104 REM *    avec la   *
105 REM * Mini-memoire *
106 REM ***************
107 REM
110 CALL CLEAR
120 CALL LOAD(-31788,227)
130 PRINT "APPUYEZ SUR UNE TOUCH
E"
140 CALL KEY(0,K,S)
150 IF K<1 THEN 140
160 CALL CLEAR
170 PRINT "D POUR ACCELERER"
180 PRINT "S POUR RALENTIR"
190 CALL CHAR(33,"7071FFE3E3FF71
7")
200 CALL CHAR(34,"")
210 CALL CHAR(35,"0080E0E0E0E08"
)
220 CALL POKEV(768,34,2,128,4,20
8)
230 CALL LOAD(-31878,1)
240 CALL POKEV(1920,0,0)
250 CALL KEY(0,K,S)
260 IF S=0 THEN 250
270 CALL PEEKV(1921,V)
280 IF K=83 THEN 350
290 IF K<>68 THEN 250
300 IF V>126 THEN 330
310 V=V+1
320 GOTO 340
330 V=127
340 GOTO 390
350 IF V<1 THEN 380
360 V=V-1
370 GOTO 390
380 V=0
390 CALL POKEV(1921,V)
400 V1=INT(V/100)
410 V2=V-100*INT(V/100)
420 V3=INT(V2/10)
430 V4=V2-10*V3
440 CALL POKEV(496,144+V1,144+V3
,144+V4)
450 GOTO 250
```

Le programme COURSE crée un lutin agrandi sur quatre caractères (codes 32, 33, 34 et 35), ce qui nécessite que vous appuyez sur n'importe quelle touche au départ.
A l'aide des touches, vous pouvez ensuite modifier la vitesse pour faire accélérer ou ralentir la voiture. Après l'arrêt, tapez CALL LOAD(−31878,0) pour faire disparaître tout lutin. A tout instant, les adresses 768 et 769 contiennent les positions verticales et horizontales du lutin (coin supérieur gauche).

Remarques complémentaires

Si vous ne possédez pas la Mini-mémoire mais le module Editeur/Assembleur, tout ce qui a été dit reste valable, sauf dans la section "Fichiers en Mini-mémoire", où le fichier devrait être créé sur disquette.

Calcul des adresses décimales

Pour convertir une adresse hexadécimale en adresse décimale utilisable pour CALL LOAD ou CALL PEEK, faire la conversion hexadécimale-décimale. Si le résultat est supérieur à 32767, soustraire 65536, sinon laisser tel quel.
Chaque adresse représente la position en mémoire d'un octet (8 bits). ■

Communiqué de Texas Instruments France

1) Texas Instruments France confirme l'arrêt de la fabrication du TI-99/4A, depuis novembre dernier.
Depuis Noël 83, suite aux ruptures de stock, Texas Instruments France s'efforce d'identifier d'éventuels stocks de matériels restant chez des revendeurs européens, et de les mettre à disposition d'autres revendeurs français en ayant fait la demande, ceci après vérifications et mises aux normes locales (manuel, transformateur d'alimentation, sortie SECAM ou Péritel).

Le consommateur pourra encore trouver quelques consoles, manettes de jeu et modules, notamment :

Mini-mémoire
Basic Etendu
Othello
Amazing
Tombstone City
Foot Ball (français)
Parsec
TI-Invaders
Blastoo
Car Wars
The Attack
Munch-Man

Quelques nouveaux logiciels, annoncés par Texas Instruments au SICOB dernier, vont quand même voir le jour :

Jawbreaker
Demon attack
Gestion privée
Mash
Hopper
Burger time
Tresure island
Star treck
Retour du pirate

Extensions périphériques

Elles sont introuvables, les possesseurs de TI-99 s'étant littéralement jetés dessus dès l'annonce de l'arrêt de la fabrication, bouleversant en cela toutes les prévisions.

2) Par ailleurs, nous pouvons confirmer que Texas Instruments France a bien porté sa garantie, sur la ligne TI-99/4A, de six mois à un an, pour les achats faits à partir du premier novembre 1983.
En outre, Texas Instruments confirme la poursuite de tous les moyens (équipe technique et pièces détachées) pour continuer le service après vente pendant plusieurs années, au-delà de la période de garantie.

***NDLR** : d'après certaines rumeurs, Texas Instruments aurait cédé plusieurs licences à des constructeurs indépendants, dans le but d'améliorer la situation des utilisateurs du TI-99/4A. Si tel est le cas, nous pouvons nous attendre à voir arriver quelques nouveaux logiciels.*
D'autre part, les modules annoncés partiront certainement très rapidement, nous ne pouvons donc que vous conseiller de "guetter" l'arrivée de ces logiciels chez votre revendeur (il serait peut-être prudent de réserver). ■

Le serveur de restaurant

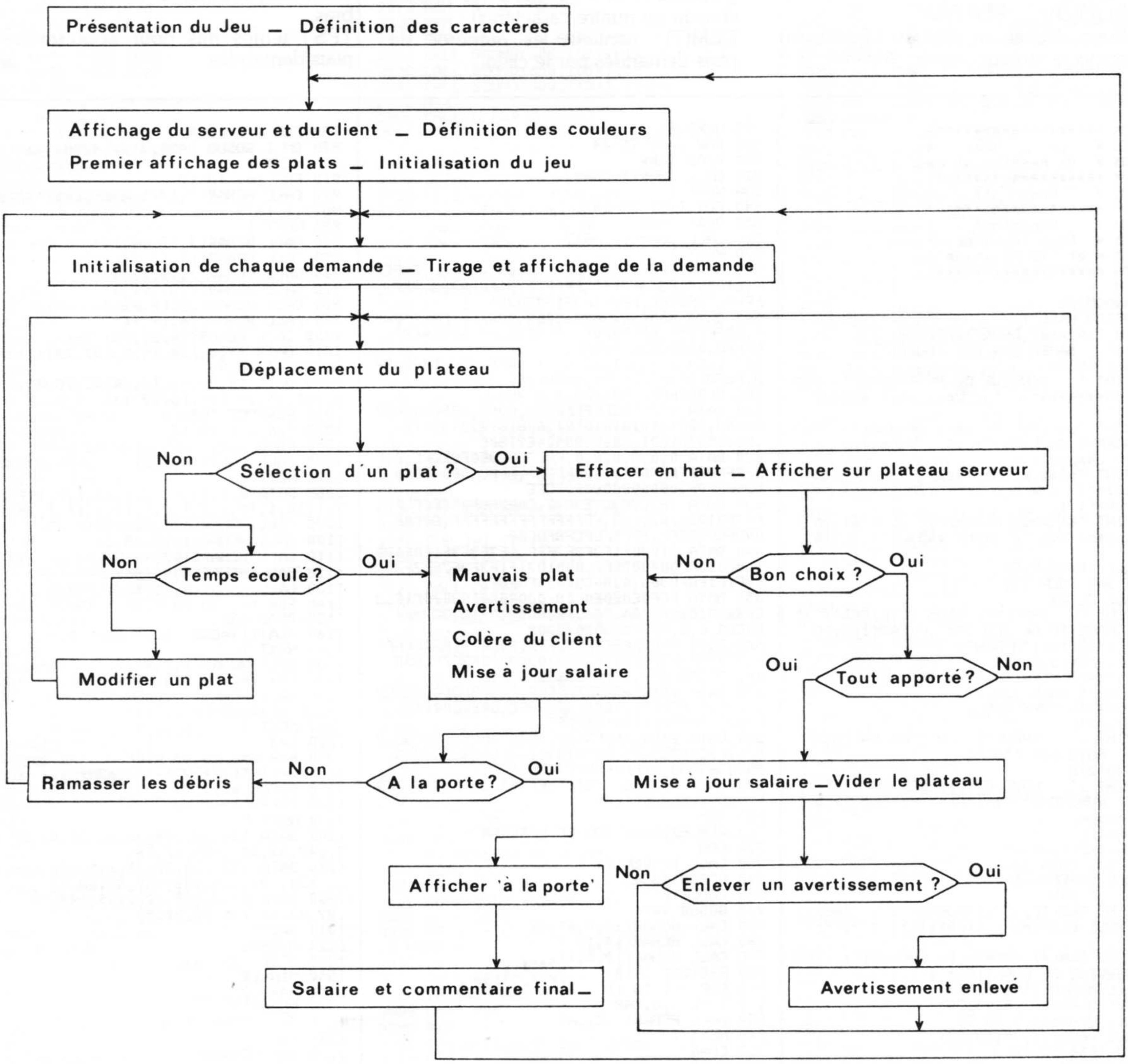
Edouard Krok

***NDLR** : ce programme occupe une grande partie de la mémoire vive, et n'est donc pas utilisable directement par les lecteurs munis d'un contrôleur de disquette (voir l'article de Denise Amrouche, dans ce numéro). Si tel est votre cas, vous devrez, avant de charger le programme en mémoire, taper CALL FILES(1) puis NEW. Il est ensuite nécessaire de supprimer toutes les lignes de REMarques. Après cela, le programme fonctionne normalement.*

Ce jeu écrit en **Basic TI** convient à tous : il suffit d'aller chercher les plats ou couverts demandés par le client. Mais bientôt, le client devient plus exigeant.

"... que m'a-t'il demandé ? De la salade ou de la purée ? Je ne me souviens plus ! Ce dont je suis sûr, c'est qu'il m'a commandé de la saucisse, mais j'ai beau guetter les plats qui arrivent de la cuisine, il n'y en a pas ! Le client va finir par se lasser d'attendre ! Peut-être se plaindra-t-il au patron du restaurant ? ... et toujours pas de saucisses !

Soudain, je les vois là-bas, au fond ! Je me précipite pour arriver avant qu'un autre serveur ne les prenne; j'attrappe ce plat tant attendu et je m'empresse de l'amener au client. Par comble de malchance, dans ma précipitation, ce n'est pas le plat de saucisses que j'ai attrapé, mais un steak qui était à côté. Cela n'a pas l'air de plaire à mon client qui se met à vociférer et entre dans une colère noire, tout en donnant de grands coups de poings sur la table qui ne tarde pas à se briser, cependant que le couvert vole en éclats. J'entends quelqu'un dans la salle siffler ironiquement : "Meunier, tu dors...", juste pour me faire remarquer que je manque de vigilance... quel métier !



Le patron, appelé, me donne un avertissement et me supprime 10% de mon salaire. Je vais essayer de me rattraper : peut-être m'enlèvera-t-il cet avertissement.
Sinon, au quatrième, je serai mis à la porte et il ne me restera plus qu'à chercher une place dans un autre restaurant."

Pour comprendre le fonctionnement du programme

Il y a 12 plats (ou couverts), composés chacun de quatre caractères graphiques, soit un total de $4 \star 12 = 48$ caractères, qui sont placés dans la matrice PLAT(48) selon un ordre précis :

1	2		5	6		9	10	→	45	46
3	4		7	8		11	12		47	48

Pour déterminer un plat, on se sert dans le programme de :
PLAT(1), PLAT(5), PLAT(9),......PLAT(45).
Pour afficher un plat sur l'écran, on respecte toujours l'ordre suivant :
3, 4, 1, 2 - 7, 8, 5, 6 - 11, 12, 9, 10..........47, 48, 45, 46.

La matrice DEM(7) sert à savoir quels sont les plats qui ont été demandés par le client, tandis que la matrice PRES(8) sert à savoir quels sont les plats qui sont présentés sur l'écran. Toutes deux ne peuvent contenir que des 0 ou l'un des 12 numéros suivants :
1 - 5 - 9 - 13 - 17 - 21 - 25 - 29 - 33 - 37 - 41 - 45.
Ces numéros sont déterminés au hasard par la formule :
(INT(12★RND)+1)★4-3.
Ils suffisent pour retrouver le plat désiré et éventuellement les graphismes qui le composent dans PLAT(No), PLAT(No+1), PLAT(No+2), PLAT(No+3).

Principales variables

PLAT(48) : contient les caractères graphiques des 12 plats (composés chacun de quatre caractères).
DEM(7) : contient les numéros de plats demandés par le client.
PRES(8) : contient les numéros des plats présents sur l'écran.
APP : compte le nombre de plats apportés.
AVER : compte le nombre d'avertissements.
CC et CL : correction colonne et correction ligne. Ces variables permettent d'afficher le serveur, "en service" et "à la porte".
MAX : nombre maximum de plats pouvant être demandés.
MIN : nombre minimum de plats pouvant être demandés.
MSAL : salaire maximum.
OT : compte le nombre de plats apportés depuis le dernier avertissement. Sert à enlever un avertissement.
RECH : contient le numéro de la colonne où se trouve le petit plateau qui va chercher les plats.
SAL : salaire du serveur.
TETE : détermine (au hasard) si le client doit tourner la tête et lever le bras.
TPS : temps mis pour apporter les plats demandés. ■

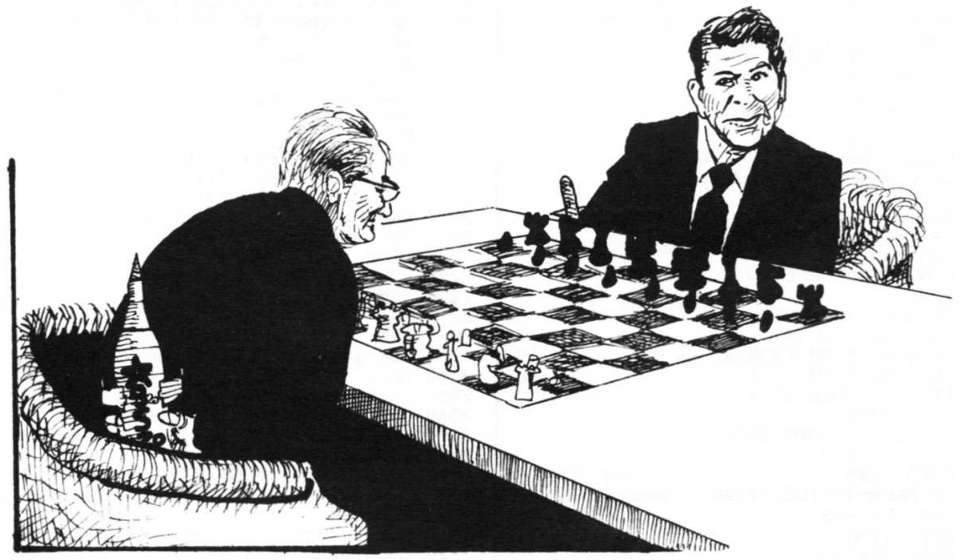
```
100 REM ******************
105 REM *   Le serveur   *
110 REM *  de restaurant  *
115 REM ******************
120 REM *    Basic TI    *
125 REM ******************
130 REM *    Copyright   *
132 REM *  Edouard Krok  *
134 REM * et "99 Magazine"*
136 REM ******************
138 REM
140 RANDOMIZE
150 OPTION BASE 1
160 DIM PLAT(48),DEM(7),PRES(8)
170 T$="   TAPEZ SUR UNE TOUCHE"
180 CALL CLEAR
190 PRINT "    SERVEUR DE RESTAURANT": :TA
B(10);"********": :"   Vous etes embauc
he pour": :
200 PRINT "assurer le service dans": :"un
 RESTAURANT de LUXE.": :"   Vous irez c
hercher chaque": :
210 PRINT "plat en deplacant le plateau":
 :TAB(8);">  : a droite": :TAB(8);"<  :
 a gauche": :
220 PRINT "et vous choisirez le plat en":
 :"appuyant sur la barre espace": : :T$
;
230 CALL KEY(3,K,S)
240 IF S=0 THEN 230
250 CALL CLEAR
260 PRINT "    PREFEREZ VOUS UTILISER": :"
   UNE MANETTE DE JEU ?": : :TAB(10);"O
_OUI": :TAB(10);"N_NON": :
270 CALL KEY(5,K,S)
280 IF (K=78)+(K=110)THEN 380
290 IF K=79 THEN 320
300 IF K=111 THEN 340
310 GOTO 270
320 PRINT :"    APPUYEZ SUR ""ALPHA LOCK""
": :"   PUIS SUR ""O"""
330 GOTO 270
340 PRINT :"    VOUS UTILISEREZ LA": :"
MANETTE NUMERO 2": : :"Appuyez sur le b
outon rouge"
350 MAN=1
360 CALL KEY(2,K,S)
370 IF S=0 THEN 360
380 CALL SCREEN(15)
390 CALL CLEAR
400 PRINT TAB(10);"RESTAURANT": : :TAB(7)
;"LA TOQUE BLANCHE": :TAB(11);"!!!!!!!!
": : : : :
410 PRINT TAB(7);"MENUS DE CHOIX": : :TAB
(7);"SERVICE DE QUALITE": : : : :
420 CALL HCHAR(3,5,45,25)
430 CALL HCHAR(22,5,45,25)
440 CALL VCHAR(3,5,45,20)
450 CALL VCHAR(3,29,45,20)
460 FOR I=34 TO 46
470 READ CAR$
480 CALL CHAR(I,CAR$)
490 NEXT I
500 FOR I=58 TO 64
510 READ CAR$
520 CALL CHAR(I,CAR$)
530 NEXT I
540 FOR I=91 TO 149
550 READ CAR$
560 CALL CHAR(I,CAR$)
570 NEXT I
580 DATA FFFFFFFFFFFFFFFF,80C0F0F8FCFCFEF
F,01030F1F3F3F7FFF,FFFF7F7F1F0F0703,FFF
FFEFEF8F0E0C,FF7F3F3F1F0F0301
590 DATA 000C3063820600C08,00FFFFFFFFFFFFF
F,00FFFFFF,0F0F0F0F0F0FFF3F,F4FAF2F4F8F
0FFFC,0000002A2A2A3E1C
600 DATA 08081C1C1C1C1C1C,FFFFFFFFF8F8F8F
8,F8F8F8F8FFFFFFFF,80008000801F8F87,000
0000000FFFFFF,0001000100F9F0E1
610 DATA EEEEEEEEEEEEEEEE,00FE,0080808080
80808,0001010101010101,0081818181818181
,008080601E01,C0600804O4E21805
620 DATA 0,0,0,0,0,0,F8F8E0C8E0F8FCFE,0,F
FFFFFFFFFFFFFFF,1F3FFFF8F,FFFC,00000000
030F3F7F,00000000C0F0FCFE
630 DATA 7E3C7E3C7E3C7E,C0E0F0F0F8FCFCFC,
FF7F010301030103,FFFFFFFFFFFFFFFF,80C0F
0F8FCFCFEFF,FFFEFCFCF8F0C08
640 DATA 01030F1F3F3F7FFF,FF7F3F3F1F0F030
1,000000000007EFF,0001071F1F3F3F7F,7F7
F3F1F1F0F0303,0080C0E0E0F0F8FC
650 DATA FCF0E0E0E0C08,0000000103070F1E,3
CF860A1030F060A,5060F0C0C5860F1C,3C78F0
E0C08,0,0,FFFF808080808080
660 DATA FFFFFFFFFFFFFFFF,80C0F0F8FCFCFEF
F,01030F1F3F3F7FFF,000103030F0F3F7F,80C
0E0F0FCFCFEFF,0818183838383818
670 DATA 1818183C7EFFFFFF,7E3C3C3C3C3C3C7
E,00FFFFFFFFFFFF,0000F0F8F8FC,0000000001
E3E7F,7FFFFFFF3F1F1F0F
680 DATA E0E0F0F0F0E0C,0000040C1C1C3C3C,7
C7C7C7C7C7C7C7C,08101020264F5F86,383838
3838383838,FFFFFFFFFF7E7E3C
690 DATA 18181818181818FF,000C0E0687E77F3
F,80C0CCCCECEFFEFE
700 REM
710 REM PREPARATION DE L'ECRAN
720 REM
730 CALL SCREEN(2)
740 GOSUB 4530
750 COUL=16
760 GOSUB 4460
770 CALL HCHAR(14,8,42,8)
780 CALL HCHAR(14,10,41,4)
790 CALL VCHAR(19,5,112,5)
800 RESTORE 1010
810 FOR I=1 TO 11
820 READ L,C,CAR
830 CALL HCHAR(L,C,CAR)
840 NEXT I
850 CC=0
860 CL=0
870 COUL=1
880 FOR I=1 TO 4
890 ON I GOSUB 3450,4100,4240,4460
900 NEXT I
910 FOR I=1 TO 12
920 CALL HCHAR(8,17+I,ASC(SEG$("APPORTEZ
MOI",I,1)))
930 NEXT I
940 CALL HCHAR(7,17,64,14)
950 CALL HCHAR(11,17,64,14)
960 CALL VCHAR(7,16,92,4)
970 CALL VCHAR(7,31,91,4)
980 CALL VCHAR(11,19,93,3)
990 CALL HCHAR(14,19,94)
1000 CALL HCHAR(14,20,95)
1010 DATA 19,6,114,24,5,137,24,6,138,13,2
3,117,14,23,98
1020 DATA 14,22,96,14,24,97,15,23,99,15,2
2,100,15,24,101,16,23,136
1030 RESTORE 1080
1040 FOR I=1 TO 15
1050 READ L,C
1060 CALL COLOR(I,L,C)
1070 NEXT I
1080 DATA 6,1,14,1,12,6,12,6,1,1,1,1,1,1,
1,1,10,1,10,1,7,1,7,1,12,1,11,1,13,1
1090 CALL HCHAR(1,1,32,192)
1100 CALL HCHAR(1,25,48,3)
1110 CALL HCHAR(1,7,48,3)
1120 CALL HCHAR(4,3,136)
1130 CALL HCHAR(4,31,136)
1140 FOR I=1 TO 48
1150 READ CAR
1160 PLAT(I)=CAR
1170 NEXT I
1180 CALL HCHAR(1,7,48,3)
1190 FOR I=1 TO LEN(STR$(MSAL))
1200 CALL HCHAR(1,9-LEN(STR$(MSAL))+I,ASC
(SEG$(STR$(MSAL),I,1)))
1210 NEXT I
1220 L=4
1230 FOR C=6 TO 27 STEP 3
1240 W=(INT(12*RND)+1)*4-3
1250 GOSUB 4590
1260 PRES(C/3-1)=W
1270 NEXT C
1280 DATA 134,32,145,32,45,32,46,32,146,3
2,147,32,107,108,37,38
1290 DATA 132,133,37,38,118,120,119,121,1
22,124,123,125,148,149,37,38
1300 DATA 36,35,39,128,139,144,140,141,32
,32,43,44,135,32,59,32
1310 SAL=0
1320 AVER=0
1330 OT=0
1340 MIN=.7
1350 MAX=.2
1360 RECH=15
1370 CALL HCHAR(5,RECH,42,2)
1380 REM
1390 REM COMMANDE DU CLIENT
1400 REM
1410 APP=0
```

```
1420 TPS=0
1430 IF MIN>6 THEN 1450
1440 MIN=MIN+.3
1450 IF MAX>6.2 THEN 1470
1460 MAX=MAX+.8
1470 QDEM=INT(((MAX-MIN+1)*RND)+MIN)
1480 FOR I=1 TO QDEM
1490 DEM(I)=(INT(12*RND)+1)*4-3
1500 NEXT I
1510 TETE=INT(6*RND)+1
1520 IF TETE<4 THEN 1540
1530 GOSUB 3470
1540 COUL=16
1550 GOSUB 4460
1560 CALL SOUND(500,800,10,-7,10)
1570 L=10
1580 FOR I=1 TO QDEM
1590 C=15+I*2
1600 W=DEM(I)
1610 GOSUB 4590
1620 NEXT I
1630 FOR DELAI=1 TO QDEM*20+50
1640 CALL KEY(2,K,S)
1650 IF S<>0 THEN 1670
1660 NEXT DELAI
1670 IF TETE<4 THEN 1690
1680 GOSUB 3660
1690 CALL HCHAR(9,17,32,14)
1700 CALL HCHAR(10,17,32,14)
1710 COUL=1
1720 GOSUB 4460
1730 CALL SOUND(200,1000,5)
1740 REM
1750 REM  DEPLACEMENT
1760 REM
1770 IF MAN THEN 1890
1780 CALL KEY(3,K,S)
1790 IF K=44 THEN 1830
1800 IF K=46 THEN 1860
1810 IF K=32 THEN 2120
1820 GOTO 2000
1830 IF RECH=6 THEN 2000
1840 RECH=RECH-3
1850 GOTO 1980
1860 IF RECH=27 THEN 2000
1870 RECH=RECH+3
1880 GOTO 1980
1890 CALL KEY(2,K,S)
1900 IF K=18 THEN 2120
1910 CALL JOYST(2,X,Y)
1920 IF X=0 THEN 2000
1930 RECH=RECH+3*X/4
1940 IF RECH>4 THEN 1960
1950 RECH=6
1960 IF RECH<28 THEN 1980
1970 RECH=27
1980 CALL HCHAR(5,3,32,29)
1990 CALL HCHAR(5,RECH,42,2)
2000 L=4
2010 C=(INT(8*RND)+1)*3+3
2020 IF C=RECH THEN 1780
2030 W=(INT(12*RND)+1)*4-3
2040 TPS=TPS+1
2050 IF TPS=QDEM*8+30 THEN 2320
2060 GOSUB 4590
2070 PRES(C/3-1)=W
2080 GOTO 1770
2090 REM
2100 REM CHOIX D'UN PLAT
2110 REM
2120 CALL SOUND(200,-5,4)
2130 IF PRES(RECH/3-1)=0 THEN 1780
2140 CALL HCHAR(3,RECH,32,2)
2150 CALL HCHAR(4,RECH,32,2)
2160 APP=APP+1
2170 IF APP>4 THEN 2210
2180 L=13
2190 C=6+APP*2
2200 GOTO 2240
2210 L=10
2220 C=APP*2-1
2230 CALL HCHAR(11,9,42,6)
2240 W=PRES(RECH/3-1)
2250 GOSUB 4590
2260 FOR I=1 TO QDEM
2270 IF DEM(I)=PRES(RECH/3-1)THEN 2570
2280 NEXT I
2290 REM
2300 REM  MAUVAIS PLAT
2310 REM
2320 CALL SOUND(300,294,3)
2330 CALL SOUND(600,392,3)
2340 CALL SOUND(300,494,3)
2350 CALL SOUND(900,392,3)
2360 PRES(RECH/3-1)=0
2370 GOSUB 3820
2380 FOR I=310 TO 110 STEP -50
2390 CALL SOUND(300,I*2,1,500,6)
2400 CALL SOUND(500,I,1,300,6)
2410 NEXT I
2420 SAL=INT(SAL*.9)
2430 OT=0
2440 GOSUB 4670
2450 AVER=AVER+1
2460 IF AVER>3 THEN 2830
2470 CALL HCHAR(1,13+AVER*2,63)
2480 FOR L=9 TO 13
2490 CALL HCHAR(L,8,32,8)
2500 NEXT L
2510 GOSUB 4070
2520 GOSUB 3450
2530 GOTO 1410
2540 REM
2550 REM  BON PLAT
2560 REM
2570 CALL SOUND(100,1000,6)
2580 PRES(RECH/3-1)=0
2590 DEM(I)=0
2600 OT=OT+1
2610 IF APP<QDEM THEN 1780
2620 REM
2630 REM  TOUT APPORTE
2640 REM
2650 CALL SOUND(300,880,1,440,4,1760,2)
2660 CALL SOUND(150,784,1,392,4,784,2)
2670 CALL SOUND(450,784,1,392,4,784,2)
2680 SAL=SAL+APP*10
2690 GOSUB 4670
2700 FOR L=9 TO 13
2710 CALL HCHAR(L,8,32,8)
2720 NEXT L
2730 IF AVER=0 THEN 1410
2740 IF OT<10 THEN 1410
2750 IF TPS>50 THEN 1410
2760 CALL HCHAR(1,13+AVER*2,32)
2770 AVER=AVER-1
2780 OT=0
2790 GOTO 1410
2800 REM
2810 REM  A LA PORTE
2820 REM
2830 FOR I=50 TO 0 STEP -50
2840 CALL SOUND(600,150+I,1)
2850 CALL SOUND(600,180+I,1)
2860 CALL SOUND(600,110+I,1)
2870 NEXT I
2880 CALL CLEAR
2890 PRINT "      VOUS ETES RENVOYE": : :
: : : : : : : : : : : : : : : : : : : : :
2900 COUL=16
2910 GOSUB 4460
2920 CALL VCHAR(4,4,34,17)
2930 CALL VCHAR(4,5,34,17)
2940 CALL VCHAR(4,13,34,17)
2950 CALL VCHAR(4,14,34,17)
2960 CALL HCHAR(4,4,34,11)
2970 CALL HCHAR(4,4,34,11)
2980 GOSUB 4370
2990 CL=-2
3000 CC=16
3010 GOSUB 4240
3020 CL=3
3030 GOSUB 4370
3040 CALL HCHAR(17,21,112,2)
3050 CALL HCHAR(18,21,116)
3060 CALL HCHAR(18,20,40)
3070 CALL HCHAR(19,19,40,2)
3080 PRINT T$
3090 CALL KEY(2,K,S)
3100 IF S=0 THEN 3090
3110 REM
3120 REM COMMENTAIRE FINAL
3130 REM
3140 FOR I=1 TO 16
3150 CALL COLOR(I,2,1)
3160 NEXT I
3170 CALL CLEAR
3180 CALL SCREEN(15)
3190 IF MSAL>SAL THEN 3210
3200 MSAL=SAL
3210 PRINT "VOTRE SALAIRE ";SAL;"FRANCS":
 : : : : :
3220 IF SAL>500 THEN 3340
3230 IF SAL>300 THEN 3320
3240 IF SAL>200 THEN 3300
3250 IF SAL>80 THEN 3280
3260 PRINT "   JE VOUS CONSEILLE": : :"
 DE CHANGER DE METIER"
3270 GOTO 3350
3280 PRINT "   VOUS AVEZ BESOIN": : :"
D APPRENDRE LE METIER"
3290 GOTO 3350
3300 PRINT "    EXERCEZ VOTRE MEMOIRE": :
:"    ET REVENEZ ME VOIR"
3310 GOTO 3350
3320 PRINT "    SI VOUS LE DESIREZ": : :"
  JE PEUX VOUS REEMBAUCHER"
3330 GOTO 3350
3340 PRINT " VOUS ETES REEMBAUCHE": : :"
JE VOUS SUPPLIE D ACCEPTER"
3350 PRINT : : : : : : : : :T$
3360 CALL KEY(2,K,S)
3370 IF S=0 THEN 3360
3380 GOTO 730
3390 REM
3400 REM AFFICHAGE DU SERVEUR
3410 REM       ET DU CLIENT
3420 REM
3430 CALL CHAR(103,"FFFFF8F8C09838F")
3440 RETURN
3450 CALL CHAR(103,"FF7FBC98C0C0F8F8")
3460 RETURN
3470 CALL CHAR(96,"01070104010F3F3F")
3480 CALL CHAR(97,"0080E060B0B0B878")
3490 CALL CHAR(98,"FFFFFFFCFBFEFFFF")
3500 CALL CHAR(100,"FF1F000001031F03")
3510 CALL CHAR(99,"DE3F7FFFFFFFFFFF")
3520 CALL CHAR(101,"F8F8F8F0F0F0C08")
3530 IF TETE<6 THEN 3650
3540 CALL VCHAR(17,20,32,3)
3550 CALL HCHAR(19,21,32)
3560 CALL HCHAR(17,21,34)
3570 CALL HCHAR(18,21,39)
3580 CALL HCHAR(16,20,35)
3590 CALL HCHAR(17,20,39)
3600 CALL HCHAR(15,19,35)
3610 CALL HCHAR(16,19,39)
3620 CALL HCHAR(15,18,39)
3630 CALL HCHAR(14,18,110)
3640 CALL HCHAR(14,17,111)
3650 RETURN
3660 IF TETE<6 THEN 3750
3670 FOR W=13 TO 16
3680 CALL VCHAR(W,W+4,32,2)
3690 NEXT W
3700 CALL HCHAR(17,21,36)
3710 CALL VCHAR(18,21,34,2)
3720 CALL HCHAR(19,20,109)
3730 CALL HCHAR(18,20,145)
3740 CALL HCHAR(17,20,134)
3750 CALL CHAR(96,"0103072E3C2E3F1F")
3760 CALL CHAR(97,"80C0E0743C74FCF8")
3770 CALL CHAR(98,"FFFFFF3C993CFFFF")
3780 CALL CHAR(99,"FFFFFF3C81C3FFFF")
3790 CALL CHAR(100,"1F1D3E3F3F1F0701")
3800 CALL CHAR(101,"F8B87CFCFCF8E08")
3810 RETURN
3820 CALL CHAR(99,"FFC300000018FFFF")
3830 CALL CHAR(100,"1F1F3F3C381C0701")
3840 CALL CHAR(101,"F8F8FC3C1C38E08")
3850 GOSUB 3430
3860 CALL VCHAR(17,20,32,3)
3870 CALL HCHAR(19,21,39)
3880 CALL HCHAR(19,22,35)
3890 CALL HCHAR(20,22,39)
3900 CALL HCHAR(20,23,35)
3910 CALL HCHAR(21,23,39)
3920 CALL HCHAR(20,17,32,3)
3930 FOR W=22 TO 18 STEP -1
3940 CALL HCHAR(W,W+1,113)
3950 CALL HCHAR(W+1,W+1,116)
3960 NEXT W
3970 CALL HCHAR(21,24,109)
3980 CALL HCHAR(19,23,40,2)
3990 CALL HCHAR(20,24,40,2)
4000 CALL HCHAR(17,28,40)
4010 CALL HCHAR(20,28,32,2)
4020 FOR W=25 TO 28
4030 CALL HCHAR(46-W,W,115)
4040 CALL HCHAR(47-W,W,114)
4050 NEXT W
4060 RETURN
4070 FOR W=23 TO 17 STEP -1
4080 CALL HCHAR(W,19,32,10)
4090 NEXT W
4100 CALL VCHAR(21,19,145,3)
4110 CALL VCHAR(21,27,145,3)
4120 CALL HCHAR(17,22,34,3)
4130 CALL HCHAR(18,21,34,5)
4140 CALL HCHAR(20,17,112,13)
4150 RESTORE 4220
4160 FOR W=1 TO 11
4170 READ L,C,CAR
4180 CALL HCHAR(L,C,CAR)
4190 NEXT W
4200 GOSUB 3700
4210 RETURN
4220 DATA 19,22,60,19,23,61,19,24,62,19,2
5,34,19,26,109,17,25,35
4230 DATA 18,26,46,17,26,45,19,28,59,18,2
8,58,17,28,135
4240 RESTORE 4340
4250 FOR W=4 TO 6
4260 CALL VCHAR(14+CL,W+CC,63,5)
4270 NEXT W
4280 CALL HCHAR(15+CL,7+CC,137,3)
4290 FOR W=1 TO 16
4300 READ L,C,CAR
4310 CALL HCHAR(L+CL,C+CC,CAR)
4320 NEXT W
4330 RETURN
4340 DATA 19,4,116,13,5,129,13,4,131,13,6
,130,12,5,136,11,5,104
4350 DATA 10,5,104,11,4,119,10,4,112,9,4,
118,9,5,112,9,6,120
4360 DATA 10,6,102,11,6,103,15,10,105,15,
11,106
4370 FOR W=14 TO 17
4380 CALL HCHAR(W+CL,W-8+CC,113)
4390 CALL HCHAR(W+1+CL,W-8+CC,112)
4400 CALL HCHAR(W+2+CL,W-8+CC,116)
4410 NEXT W
```

```
4420 CALL HCHAR(19+CL,9+CC,32)
4430 CALL VCHAR(17+CL,10+CC,143,2)
4440 CALL HCHAR(16+CL,10+CC,142)
4450 RETURN
4460 FOR W=5 TO 8
4470 CALL COLOR(W,COUL,1)
4480 NEXT W
4490 RETURN
4500 REM
4510 REM MESSAGE D'ATTENTE
4520 REM
4530 CALL CLEAR
4540 PRINT TAB(7);"TENEZ VOUS PRET": : :"
   LES PREMIERS CLIENTS": :"   NE DEVRA
IENT PAS TARDER": : : : : : : : : : : : :
 : : : : :
4550 RETURN
4560 REM
4570 REM AFFICHAGE DES PLATS
4580 REM
4590 CALL HCHAR(L,C,PLAT(W+2))
4600 CALL HCHAR(L,C+1,PLAT(W+3))
4610 CALL HCHAR(L-1,C,PLAT(W))
4620 CALL HCHAR(L-1,C+1,PLAT(W+1))
4630 RETURN
4640 REM
4650 REM AFFICHAGE DU SALAIRE
4660 REM
4670 CALL HCHAR(1,25,48,3)
4680 FOR W=1 TO LEN(STR$(SAL))
4690 FOR I=48 TO ASC(SEG$(STR$(SAL),W,1))
4700 CALL HCHAR(1,27-LEN(STR$(SAL))+W,I)
4710 CALL SOUND(100,W*I*10,5)
4720 NEXT I
4730 NEXT W
4740 RETURN
```

Le module "Echecs" à l'essai

Alain Paloma



Dans la catégorie des jeux de réflexion, Texas Instruments commercialise sous forme de module un magnifique programme de jeu d'échecs.

Ma première surprise fut de constater qu'après avoir introduit le module, l'habituelle mire indiquant :

READY - PRESS ANY KEY TO BEGIN

était devenue :

POUR COMMENCER APPUYEZ SUR N'IMPORTE QUELLE TOUCHE

L'écran suivant est aussi en français :

ORDINATEUR FAMILIAL
CHOISISSEZ :
1-UTILISATION EN "BASIC"
2-CALCUL D'UNE EQUATION
3-JEU D'ECHECS

(Que vient donc faire ici le calcul d'une équation, serait-ce un nouveau "bug" ?).

Ceci n'a certes que peu de rapport avec les échecs, mais méritait cependant d'être souligné.

Venons-en au programme d'échecs proprement dit. Avant d'aller plus loin dans l'exploration de ce logiciel, mettons les choses au point. Ce module permet aux débutants de s'initier rapidement et efficacement aux échecs et aux joueurs confirmés d'avoir, toujours disponible, un partenaire de choix, mais qui ne peut en aucune façon mettre sérieusement en danger un adversaire de haut niveau. Au niveau 2 (il y en a 3 au total), un joueur moyen (en l'occurence votre dévoué serviteur) peut arriver à le battre (je n'y suis jamais parvenu au niveau supérieur). Ceci étant dit, jouons...

Une fois de plus, le graphisme est très soigné (nous avions fait la même remarque pour le module "Othello"), les pièces sont finement dessinées et les couleurs assez judicieusement choisies. Une petite musique d'accompagnement se fait entendre à chaque déplacement d'une pièce (les pièces glissent sur l'échiquier, ce qui laisse supposer qu'il s'agit de "lutins").

A gauche de l'échiquier apparaissent deux chronomètres, qui comptabilisent les temps de réflexion des deux adversaires. Il est possible de jouer contre un "humain", le TI-99 étant alors réduit à la seule fonction d'échiquier, ou contre lui; il fait alors preuve, selon le choix effectué en début de partie, d'un comportement agressif, défensif ou normal. Le temps de réflexion de l'ordinateur est directement lié au niveau de jeu.

La principale attraction de ce module, outre le fait qu'il fournit un bon adversaire, est qu'il comporte une option permettant au débutant de s'initier à la stratégie des échecs.

Les options de jeu sont nombreuses : il est, par exemple, possible de permuter l'échiquier (le TI-99 prend votre jeu et vous le sien) à tout moment. La partie étant intégralement mémorisée, il est également possible de revenir en arrière d'autant de phases de jeu qu'on le désire, et de reprendre la partie (ceci est très utile quand la situation est désespérée, pour reculer l'échéance fatale).

On peut évidemment soumettre un problème à l'ordinateur, lui demander conseil en cours de partie, etc... On peut même jouer plusieurs parties simultanées : l'ordinateur s'occupe de tout.

Enfin, lorsque vous saurez que l'on peut sauvegarder (sur cassette exclusivement) une partie en cours, afin de pouvoir la reprendre ultérieurement, vous pourrez vous précipiter chez votre revendeur. ■

Remise à zéro des variables en Pascal

En Pascal, comme en Assembleur (et contrairement au Basic), lors de l'exécution d'un programme, aucune mémoire n'est remise à zéro. Cette propriété, qui est souvent très intéressante (notamment pour situer les erreurs dues aux débordements), peut parfois provoquer des erreurs pour le moins inattendues si l'on conserve ses vieux automatismes "Basiciens". N'oubliez donc jamais d'initialiser toutes vos variables, et particulièrement les tableaux, qui sont généralement les premières victimes. ■

Combat de chars

Jean-Claude Hoet

C'est à la vue de la cassette de jeu "Blastoo" que l'envie de créer un tel programme m'est venue.
Vous verrez que le jeu est très différent de son grand frère, mais que son principal atout réside dans le réalisme de l'action et dans l'intense tension que ce jeu génère chez les deux joueurs, l'ordinateur se contentant de fournir le décor de la bataille.

Comment fonctionne "Combat de chars" ?

Tout d'abord, le charger; et pour cela la configuration de base suffit puisqu'il est écrit en Basic TI.
Le contrôle des chars s'effectue à partir du clavier.

Après un court "copyright" destiné à occuper l'écran pendant la définition des caractères, une ville se dessine puis un "bip" sonore annonce le début des hostilités.
Au départ, les deux chars se trouvent respectivement en bas à gauche et à droite de l'écran.

Le char de gauche se commande avec les touches A, S, X et D, et le char de droite avec les touches H, J, M et K.
La touche A (H) permet de faire avancer le char considéré.
La touche S (J) permet de faire tourner le char dans le sens des aiguilles d'une montre.
La touche X (M) permet de faire tourner le char dans le sens inverse des aiguilles d'une montre.
Enfin, la touche D (K) permet le tir du canon.

Une certaine dextérité sera nécessaire avant de pouvoir guider son char sans encombre. Pour ma part, je conseille vivement la technique suivante :

Le majeur de la main gauche sur A (H), l'annulaire de la main gauche alternativement sur S (J) ou X (M) suivant la direction à prendre. Enfin, l'annulaire de la main droite sur la touche de tir D (K).

Ceux qui possèdent des manettes de jeu peuvent facilement modifier le programme pour une utilisation plus aisée, mais attention, le piment de ce jeu réside, entre autre, dans le fait qu'un char ne fait pas demi-tour instantanément (il lui faut quatre temps), ce qui permet à un adversaire qui parvient à ce placer derrière un char de faire feu ! (ce qui ne prend qu'un temps). Il faut qu'il en soit de même après transformation.

Vous remarquerez très vite, à vos dépends, qu'il faut deux obus pour perforer un mur intérieur et un seul si vous êtes directement contre.
Le mur extérieur à la ville est "indestructible" (les obus explosent au pied). Si vous tirez dessus à courte distance, votre char explose !

La simulation des explosions comprend une flamme à la sortie du canon et un caractère représentant une explosion. Ce caractère se place sur le premier obstacle rencontré et une simulation visuelle est obtenue en faisant varier alternativement et très rapidement la couleur du caractère et de son fond (rouge sur blanc). Un char touché provoque un véritable cataclysme.

La pression d'une touche permet ensuite d'afficher le tableau de chasse. Pour "avoir" régulièrement votre adversaire, une stratégie sans cesse renouvelée vous sera nécessaire, ce qui rend ce jeu très prenant, d'après les multiple amis qui ont eu l'occasion de s'y essayer.

Comment fonctionne ce programme ?

La limitation pour un tel jeu, écrit en **Basic TI**, est la vitesse d'exécution. Bien qu'il s'agisse ici d'un combat de chars et qu'un char soit plutôt lent, il m'a fallu accélérer au maximum le processus de déplacement des chars.

Ayant remarqué auparavant que les GOSUB et les boucles FOR-NEXT ralentissent considérablement le programme, j'ai opté pour une programmation "en ligne" où j'utilise au maximum les instructions ON-GOTO et GOTO, pour diriger le programme vers les parties qui traitent le mouvement demandé.

Il y a huit directions possibles par char, ce qui nous donne seize caractères (deux chars de couleurs différentes), huit caractères "flamme" et cinq caractères spéciaux (explosion, mur, ...).

136 à 143 : char blanc
144 à 151 : char noir
152 à 159 : flamme au canon
45 : explosion
46 : croix de la tombe
128 : mur
129 : mur extérieur
130 : débris

Le plan de la ville est fixé dans les lignes de DATA en fournissant les coordonnées nécessaires pour les instructions HCHAR et VCHAR (ligne, colonne, répétition). Le signe moins (-) dans les DATA indique au programme qu'il s'agit de données pour une instruction VCHAR, alors que l'absence du signe oriente les données vers une instruction HCHAR.

En gros, et pour le char blanc :

810 : entrée d'une commande. Si rien, GOTO 2080 (char noir)
820 : mouvement du char blanc
850 : le char blanc contre un mur (ne peut pas le traverser)
870 : efface le char de son ancienne position
900 : positionne le char à sa nouvelle position
1500 : tir du char blanc
1510 : qu'y a-t-il devant le char blanc ? (après le tir)
1590 : objectif atteint. Dessin de l'explosion puis de débris (ou de tombe)
1620 : explosion d'un obus sur un mur neuf
1640 : simulation de l'explosion
1700 : dessin de débris
1710 : efface la flamme du canon
1730 : explosion contre le mur extérieur
1840 : explosion d'un mur déjà touché et création d'un passage
1950 : explosion du char touché
1990 : simulation de l'explosion du char
2080 : même suite logique pour le char noir

En fin de programme se trouve la sous-routine qui permet d'afficher le tableau de chasse.
Ce programme occupe 7928 octets (sans les REMs) et 8413 après les réservations des variables. ■

```
10 REM ********************
20 REM * Combat de Chars  *
30 REM ********************
40 REM *     Basic TI     *
50 REM ********************
60 REM *    Copyright     *
70 REM * Jean-Claude Hoet *
80 REM * et "99 Magazine" *
90 REM ********************
100 CALL CLEAR
110 CALL SCREEN(6)
120 GOSUB 3570
125 REM **************
126 REM DEF. CARACTERES
127 REM **************
130 CALL CHAR(128,"FFFFFFFFFFFFFFFF")
140 CALL CHAR(45,"105A3E67E67C3A48")
150 CALL CHAR(130,"BC3D01CDCCC303F0")
```

```
160 CALL CHAR(129,"FFFFC3C3C3C3FFFF")
170 CALL CHAR(46,"10107C10101038FE")
180 CALL COLOR(13,11,1)
190 DATA 11,2,5,13,2,5,14,2,5,17,3,4,4,9,
2,12,9,4,14,8,4,17,8,4,3,12,7,18,15,4,2
0,13,6
200 DATA 7,21,6,14,19,6,16,22,2,17,22,2,4
,25,6,5,25,6,17,26,6,-1,4,9,-20,4,5,-17
,6,4
210 DATA -1,6,7,-1,7,7,-14,8,3,-4,9,7,-15
,11,2,-19,10,6,-4,12,4,-9,12,3,-14,13,6
220 DATA -5,15,8,-15,15,3,-4,18,8,-8,21,4
,-1,22,5,-1,23,5,-9,24,5,-20,22,5,-20,2
3,5
230 DATA -18,26,3,-20,30,5,9,26,5,13,26,5
,-10,26,3,-10,30,3,11,31,1,0,0,0
240 CALL CHAR(136,"101010387C387C38")
250 CALL CHAR(152,"0000AA2854102810")
260 CALL CHAR(153,"2018361D36182000")
270 CALL CHAR(154,"1028546CAA540000")
280 CALL CHAR(155,"04083C78A4782C08")
290 CALL CHAR(156,"008040E070F8F4AA")
300 CALL CHAR(157,"AA7CF870E0408000")
310 CALL CHAR(158,"553E5F0E17020500")
320 CALL CHAR(159,"000502170E5F3E55")
330 CALL CHAR(144,"101010387C387C38")
340 CALL CHAR(137,"00000A1FFF1F0A00")
350 CALL CHAR(145,"00000A1FFF1F0A00")
360 CALL CHAR(138,"387C387C38101010")
370 CALL CHAR(146,"387C387C38101010")
380 CALL CHAR(139,"000050F8FFF85000")
390 CALL CHAR(140,"8054387D3E5F0E14")
400 CALL CHAR(141,"012A1CBE7CFA7028")
410 CALL CHAR(142,"140E5F3E7D385480")
420 CALL CHAR(143,"2870FA7CBE1C2A01")
430 CALL CHAR(147,"000050F8FFF85000")
440 CALL CHAR(148,"8054387D3E5F0E14")
450 CALL CHAR(149,"012A1CBE7CFA7028")
460 CALL CHAR(150,"140E5F3E7D385480")
470 CALL CHAR(151,"2870FA7CBE1C2A01")
480 CALL COLOR(14,16,1)
490 CALL COLOR(15,2,1)
500 CALL COLOR(16,9,1)
510 CALL CLEAR
515 REM ************
516 REM DESSIN VILLE
517 REM ************
520 RESTORE
530 READ A,B,C
540 IF A=0 THEN 610
550 CALL SOUND(-50,ABS(A*110),2)
560 IF A<0 THEN 590
570 CALL HCHAR(A,B,128,C)
580 GOTO 530
590 CALL VCHAR(-A,B,128,C)
600 GOTO 530
610 CALL HCHAR(1,2,129,31)
620 CALL HCHAR(24,2,129,31)
630 CALL VCHAR(1,2,129,24)
640 CALL VCHAR(1,32,129,24)
650 CALL SOUND(500,500,0)
655 REM **************
656 REM INITIALISATION
657 REM **************
660 A=23
670 D=23
680 DD=-1
690 AA=-1
700 B=3
710 E=31
720 EE=0
730 BB=0
740 C=136
750 F=144
760 G(1)=152
770 G(2)=152
780 CALL HCHAR(D,E,144)
790 CALL HCHAR(A,B,136)
794 REM *************
795 REM ACTION CHAR 1
796 REM *************
800 CALL KEY(1,ASA,POL)
810 IF (POL=0)+(ASA>3)THEN 2080
820 IF ASA<>1 THEN 920
830 CALL GCHAR(A+AA,B+BB,TE)
840 IF TE<128 THEN 870
850 CALL SOUND(200,1000,4)
860 GOTO 2080
870 CALL HCHAR(A,B,127)
880 A=A+AA
890 B=B+BB
900 CALL HCHAR(A,B,C)
910 GOTO 2080
920 IF ASA=3 THEN 1500
930 IF ASA=2 THEN 950
940 ON AA+2 GOTO 960,1160,1300
950 ON AA+2 GOTO 970,1170,1310
960 ON BB+2 GOTO 1320,1180,980
970 ON BB+2 GOTO 980,1040,1100
980 AA=-1
990 BB=0
1000 G(1)=152
1010 C=136
1020 CALL HCHAR(A,B,C)
1030 GOTO 2080
1040 AA=-1
1050 BB=1
1060 G(1)=156
1070 C=141
1080 CALL HCHAR(A,B,C)
1090 GOTO 2080
1100 AA=0
1110 BB=1
1120 G(1)=155
1130 C=139
1140 CALL HCHAR(A,B,C)
1150 GOTO 2080
1160 ON BB+2 GOTO 1380,2080,1040
1170 ON BB+2 GOTO 1180,2080,1240
1180 AA=-1
1190 BB=-1
1200 G(1)=159
1210 C=140
1220 CALL HCHAR(A,B,C)
1230 GOTO 2080
1240 AA=1
1250 BB=1
1260 G(1)=157
1270 C=143
1280 CALL HCHAR(A,B,C)
1290 GOTO 2080
1300 ON BB+2 GOTO 1440,1240,1100
1310 ON BB+2 GOTO 1320,1380,1440
1320 AA=0
1330 BB=-1
1340 G(1)=153
1350 C=137
1360 CALL HCHAR(A,B,C)
1370 GOTO 2080
1380 AA=1
1390 BB=-1
1400 G(1)=158
1410 C=142
1420 CALL HCHAR(A,B,C)
1430 GOTO 2080
1440 AA=1
1450 BB=0
1460 G(1)=154
1470 C=138
1480 CALL HCHAR(A,B,C)
1490 GOTO 2080
1500 CALL SOUND(300,-5,0)
1510 CALL GCHAR(A+AA,B+BB,TE)
1520 L=A
1530 LL=B
1540 IF TE=129 THEN 3230
1550 L=L+AA
1560 LL=LL+BB
1570 CALL GCHAR(L,LL,TE)
1580 IF TE<128 THEN 1550
1590 CALL HCHAR(A+AA,B+BB,G(1))
1600 IF TE>135 THEN 1950
1610 ON TE-127 GOTO 1620,1730,1840
1620 CALL SOUND(1000,-6,0)
1630 CALL HCHAR(L,LL,45)
1640 FOR PO=1 TO 5
1650 CALL COLOR(2,9,16)
1660 FOR PA=1 TO 30
1670 NEXT PA
1680 CALL COLOR(2,16,9)
1690 NEXT PO
1700 CALL HCHAR(L,LL,130)
1710 CALL HCHAR(A+AA,B+BB,127)
1720 GOTO 2080
1730 CALL SOUND(500,-6,0)
1740 CALL HCHAR(L-AA,LL-BB,45)
1750 FOR PO=1 TO 5
1760 CALL COLOR(2,9,16)
1770 FOR PA=1 TO 30
1780 NEXT PA
1790 CALL COLOR(2,16,9)
1800 NEXT PO
1810 CALL HCHAR(L-AA,LL-BB,127)
1820 CALL HCHAR(A+AA,B+BB,127)
1830 GOTO 2080
1840 CALL SOUND(1000,-6,3)
1850 CALL HCHAR(L,LL,45)
1860 FOR PO=1 TO 5
1870 CALL COLOR(2,9,16)
1880 FOR PA=1 TO 30
1890 NEXT PA
1900 CALL COLOR(2,16,9)
1910 NEXT PO
1920 CALL HCHAR(L,LL,127)
1930 CALL HCHAR(A+AA,B+BB,127)
1940 GOTO 2080
1950 CALL SOUND(3000,-6,0)
1960 XZ=XZ+1
1970 X=1
1980 CALL HCHAR(L,LL,45)
1990 FOR PA=1 TO 5
2000 FOR PO=1 TO 16
2010 CALL SCREEN(PO)
2020 NEXT PO
2030 NEXT PA
2040 CALL SCREEN(6)
2050 CALL HCHAR(A+AA,B+BB,127)
2060 CALL HCHAR(L,LL,46)
2070 GOTO 3350
2075 REM *************
2076 REM ACTION CHAR 2
2077 REM *************
2080 CALL KEY(2,ASA,POL)
2090 IF (POL=0)+(ASA>3)THEN 800
2100 IF ASA<>1 THEN 2200
2110 CALL GCHAR(D+DD,E+EE,TE)
2120 IF TE<128 THEN 2150
2130 CALL SOUND(200,1000,4)
2140 GOTO 800
2150 CALL HCHAR(D,E,127)
2160 D=D+DD
2170 E=E+EE
2180 CALL HCHAR(D,E,F)
2190 GOTO 800
2200 IF ASA=3 THEN 2780
2210 IF ASA=2 THEN 2230
2220 ON DD+2 GOTO 2240,2440,2580
2230 ON DD+2 GOTO 2250,2450,2590
2240 ON EE+2 GOTO 2600,2460,2260
2250 ON EE+2 GOTO 2260,2320,2380
2260 DD=-1
2270 EE=0
2280 G(2)=152
2290 F=144
2300 CALL HCHAR(D,E,F)
2310 GOTO 800
2320 DD=-1
2330 EE=1
2340 G(2)=156
2350 F=149
2360 CALL HCHAR(D,E,F)
2370 GOTO 800
2380 DD=0
2390 EE=1
2400 G(2)=155
2410 F=147
2420 CALL HCHAR(D,E,F)
2430 GOTO 800
2440 ON EE+2 GOTO 2660,2080,2320
2450 ON EE+2 GOTO 2460,800,2520
2460 DD=-1
2470 EE=-1
2480 G(2)=159
2490 F=148
2500 CALL HCHAR(D,E,F)
2510 GOTO 800
2520 DD=1
2530 EE=1
2540 G(2)=157
2550 F=151
2560 CALL HCHAR(D,E,F)
2570 GOTO 800
2580 ON EE+2 GOTO 2720,2520,2380
2590 ON EE+2 GOTO 2600,2660,2720
2600 DD=0
2610 EE=-1
2620 G(2)=153
2630 F=145
2640 CALL HCHAR(D,E,F)
2650 GOTO 800
2660 DD=1
2670 EE=-1
2680 G(2)=158
2690 F=150
2700 CALL HCHAR(D,E,F)
2710 GOTO 800
2720 DD=1
2730 EE=0
2740 G(2)=154
2750 F=146
2760 CALL HCHAR(D,E,F)
2770 GOTO 800
2780 CALL SOUND(300,-5,0)
2790 CALL GCHAR(D+DD,E+EE,TE)
2800 L=D
2810 LL=E
2820 IF TE=129 THEN 1950
2830 L=L+DD
2840 LL=LL+EE
2850 CALL GCHAR(L,LL,TE)
2860 IF TE<128 THEN 2830
2870 CALL HCHAR(D+DD,E+EE,G(2))
2880 IF TE>135 THEN 3230
2890 ON TE-127 GOTO 2900,3010,3120
2900 CALL SOUND(1000,-6,0)
2910 CALL HCHAR(L,LL,45)
2920 FOR PO=1 TO 5
2930 CALL COLOR(2,9,16)
2940 FOR PA=1 TO 30
2950 NEXT PA
2960 CALL COLOR(2,16,9)
2970 NEXT PO
2980 CALL HCHAR(L,LL,130)
2990 CALL HCHAR(D+DD,E+EE,127)
3000 GOTO 800
3010 CALL SOUND(500,-6,0)
3020 CALL HCHAR(L-DD,LL-EE,45)
3030 FOR PO=1 TO 5
3040 CALL COLOR(2,9,16)
```

```
3050 FOR PA=1 TO 30
3060 NEXT PA
3070 CALL COLOR(2,16,9)
3080 NEXT PO
3090 CALL HCHAR(L-DD,LL-EE,127)
3100 CALL HCHAR(D+DD,E+EE,127)
3110 GOTO 800
3120 CALL SOUND(1000,-6,3)
3130 CALL HCHAR(L,LL,45)
3140 FOR PO=1 TO 5
3150 CALL COLOR(2,9,16)
3160 FOR PA=1 TO 30
3170 NEXT PA
3180 CALL COLOR(2,16,9)
3190 NEXT PO
3200 CALL HCHAR(L,LL,127)
3210 CALL HCHAR(D+DD,E+EE,127)
3220 GOTO 800
3230 CALL SOUND(3000,-6,0)
3240 ZX=ZX+1
3250 XX=1
3260 CALL HCHAR(L,LL,45)
3270 FOR PA=1 TO 5
3280 FOR PO=1 TO 16
3290 CALL SCREEN(PO)
3300 NEXT PO
3310 NEXT PA
3320 CALL SCREEN(6)
3330 CALL HCHAR(D+DD,E+EE,127)
3340 CALL HCHAR(L,LL,46)
3345 REM *********
3346 REM RESULTATS
3347 REM *********
3350 CALL KEY(0,ASA,POL)
3360 IF POL=0 THEN 3350
3370 CALL CLEAR
3380 CALL COLOR(2,2,1)
3390 IF X=1 THEN 3530
3400 QW$="NOIRS"
3410 QZ$="BLANCS"
3420 XX=0
3430 PRINT "LES "&QW$&" SONT LES MAITRES
 DE LA VILLE!": :"LES "&QZ$&" VEULENT I
LS UNE   AUTRE CHANCE (O/N)?": :
3440 PRINT "TABLEAU DE CHASSE:":"********
*********": :"BLANC:": :"NOIR :": : :
:
3450 CALL HCHAR(18,9,144,XZ)
3460 CALL HCHAR(20,9,136,ZX)
3470 CALL KEY(0,ASA,POL)
3480 IF POL=0 THEN 3470
3490 IF ASA<>78 THEN 510
3500 CALL CLEAR
3510 PRINT "DANS CE CAS JE DOIS EN      C
ONCLURE QUE LES "&QZ$&" SONT DES POULES
 MOUILLES": : : : : :
3520 END
3530 QW$="BLANCS"
3540 QZ$="NOIRS"
3550 X=0
3560 GOTO 3430
3570 PRINT "COMBAT DE CHARS": :"OCTOBRE 1
983": :"99 MAGAZINE ET":"HOET JEAN-CLAU
DE": : : : :
3580 RETURN
```

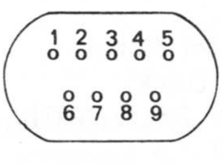
Réalisez votre cordon de magnétophone

Guy Fix

Vous êtes nombreux à ne pas pouvoir vous procurer le cordon qui permet de sauvegarder des programmes sur cassettes. Si le bricolage ne vous fait pas peur, les renseignements que nous allons vous donner pourrons vous être utiles.

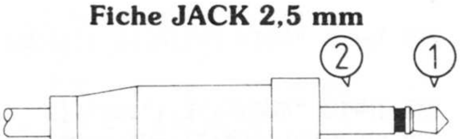
Le raccordement au TI se fait par l'intermédiaire d'une prise CANON 9 points ou SOURIAU DEF-9S.

Prise chassis



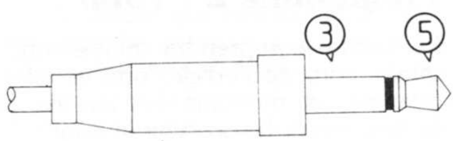
1 et 2 : télécommande moteur CS1

Fiche JACK 2,5 mm



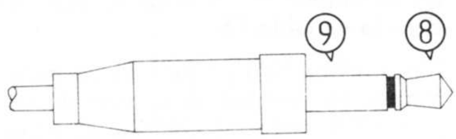
3 et 5 : sortie magnéto CS1

Fiche JACK 3,5 mm



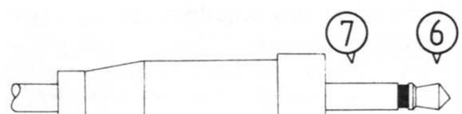
8 et 9 : entrée magnéto CS1

Fiche JACK 3,5 mm



6 et 7 : télécommande moteur CS2

Fiche JACK 2,5 mm



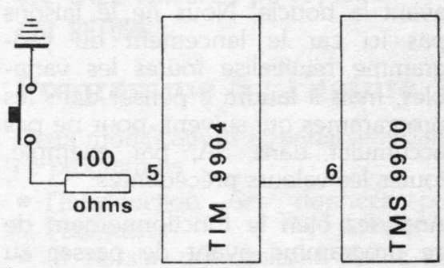
En cas de non-fonctionnement de la télécommande, inversez les fils au niveau de la fiche jack télécommande CS1 (ou CS2).

Pour le cordon magnétophone simple CS1, il ne faut pas brancher les fils télécommande moteur CS2 6 et 7.
La sortie magnétophone CS2 est en parallèle sur la sortie CS1 (3 et 5).
3 et 5 correspondent à la fiche du fil rouge (micro) du manuel TI.
8 et 9 correspondent à la fiche du fil blanc (écouteur ou haut parleur). ■

Une amélioration de la touche "RESET"

Robert Thibodeau - Québec

J'ai fait la modification pour le "RESET" et cela fonctionne parfaitement, sauf dans le cas où un synthétiseur de voix est raccordé au TI-99 car, avec cette configuration, il est impossible de reprendre le contrôle après avoir effectué un "RESET". Je me suis penché sur le problème et en suis arrivé à découvrir la solution suivante : au lieu de brancher le bouton sur la patte 6 du TMS 9900, il est possible de se placer sur la patte 5 du TIM 9904 situé juste au-dessus du TMS 9900, en ajoutant une résistance de 100 ohms.



Basic Etendu vers Basic TI

Lorsque l'on charge un programme écrit en Basic TI alors que l'on a choisi l'option 2 du menu principal, le programme ne fonctionne pas, du fait des différences entre les deux langages. Normalement, il faut à nouveau charger le programme, en Basic TI cette fois.

Nous pouvons éviter ces manipulations en utilisant la méthode suivante :

Entrez les lignes

1 !@P−
2 CALL COLOR

En faisant "RUN", vous obtenez un message d'erreur mais vous êtes passé en Basic TI. Il ne vous reste plus qu'à supprimer les lignes 1 et 2 et le programme fonctionnera normalement.

Apprendre le Basic en programmant

Roger Didi

Rappelons d'abord, pour le lecteur qui ne connaît pas notre méthode de travail, que nous construisons progressivement un programme en profitant des différentes étapes de cette construction pour préciser certaines notions.
Aujourd'hui, un client nous demande de lui construire un programme de facturation. Nous allons donc commencer par dessiner un tableau sur l'écran, puis nous apprendrons à y inscrire des données. Enfin, nous apprendrons à modifier ces données en fonction des décisions de l'utilisateur.
En ce qui concerne le Basic, nous profiterons de l'évolution de ce programme pour mieux comprendre l'utilisation des variables, et plus particulièrement des tableaux de variables. Nous perfectionnerons aussi notre approche des sous-programmes.

Programme 1 : Calculs

Ce premier programme est un essai qui permet d'observer les calculs utilisant des tableaux. A() et B() sont des tableaux à une ligne (ou une colonne) pouvant contenir jusqu'à 15 nombres (ligne 100).

SA contient la somme de tous les nombres contenus dans le tableau A().
SB contient la somme de tous les nombres contenus dans le tableau B().
P contient la somme de tous les produits des nombres de même numéro du tableau A() et du tableau B().

Normalement, il faudrait initialiser à zéro les contenus de SA, SB et P avant la boucle. Nous ne le faisons pas ici car le lancement du programme réinitialise toutes les variables, mais il faudra y penser dans les programmes qui suivent, pour ne pas accumuler dans SA, par exemple, toutes les valeurs précédentes.

Analysez bien le fonctionnement de ce programme avant de passer au suivant.

```
10 REM *****************
15 REM * PROGRAMME No 1 *
20 REM *    CALCULS     *
25 REM *****************
30 REM *   BASIC TI     *
35 REM *****************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM *****************
60 REM
100 DIM A(15),B(15)
110 FOR I=1 TO 15
120 INPUT A(I),B(I)
130 SA=SA+A(I)
140 SB=SB+B(I)
150 P=P+A(I)*B(I)
160 PRINT "SA = ";SA:"SB = ";SB:
"P = ";P
170 NEXT I
180 PRINT "SA = ";SA:"SB = ";SB:
"P = ";P
```

Programme 2 : Total

La facture comprendra quatre rubriques : nom de l'article, prix unitaire, quantité, et montant; en bas de la facture, nous devons voir le total.

Nous prévoyons donc quatre tableaux : ART$(), PU$(), QU$(), et PR$(). Chacun de ces tableaux peut contenir dix mots (ligne 120).
Nous créons d'autre part le caractère 123 qui permettra de dessiner des lignes verticales, et nous le plaçons dans la variable T$.

L'instruction VAL() permet de transformer en nombre la chaîne de caractères formée de chiffres, et écrite entre parenthèses. L'instruction STR$() accomplit l'action inverse.

Certains se demanderont pourquoi nous n'avons pas choisi directement des variables numériques pour enregistrer les nombres. L'une des raisons que nous pouvons invoquer est la crainte d'une erreur de manipulation de l'utilisateur. En effet, sur un INPUT X, par exemple, si l'utilisateur introduit autre chose qu'un chiffre, la machine affiche un message d'erreur (en anglais !) qui pourrait l'affoler... Une autre raison est l'homogénéité des variables, et nous en verrons l'intérêt plus loin.
Nous en trouvons d'ailleurs un exemple à la ligne 330 où nous voyons dans le même tableau une chaîne de caractères et des nombres.

En dehors de ces remarques, ce programme ne doit pas présenter trop de difficulté; faites-le fonctionner plusieurs fois et faites un bilan de tous ses défauts et des améliorations que vous envisagez, avant de passer au suivant.

```
10 REM *****************
15 REM * PROGRAMME No 2 *
20 REM *     TOTAL      *
25 REM *****************
30 REM *   BASIC TI     *
35 REM *****************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM *****************
60 REM
100 CALL CLEAR
110 CALL SCREEN(6)
120 DIM ART$(10),PU$(10),QU$(10)
,PR$(10)
130 ART$(0)="ARTICLE"
140 PU$(0)="P.U"
150 QU$(0)="QTE."
160 PR$(0)="A PAYER"
170 CALL CHAR(123,"1010101010101
010")
180 T$=CHR$(123)
190 INPUT "NOMBRE D'ARTICLES  ":
N
200 FOR I=1 TO N
210 CALL CLEAR
220 PRINT TAB(9);"ARTICLE No ";I
230 PRINT : : :
240 INPUT "DESIGNATION :":ART$(I
)
250 INPUT "PRIX UNITAIRE :":PU$(
I)
260 INPUT "QUANTITE :":QU$(I)
270 PRIX=VAL(PU$(I))*VAL(QU$(I))
280 TOTAL=TOTAL+PRIX
290 PR$(I)=STR$(PRIX)
300 NEXT I
310 CALL CLEAR
320 FOR I=0 TO N
330 PRINT TAB(2);ART$(I);TAB(9);
T$;TAB(10);PU$(I);TAB(15);T$;TAB
(16);QU$(I);TAB(
20);T$;TAB(21);PR$(I)
340 REM La tabulation permet une
    presentation en tableau
350 NEXT I
360 PRINT : :TAB(15);"TOTAL : ";
TOTAL
370 GOTO 370
380 REM
390 REM Pour arreter, faire FCTN
 4
400 REM
```

Programme 3 : Tableau

Nous nous intéressons dans ce programme à une amélioration de la présentation de la facture.
Les traits doivent être mieux dessinés et une flèche que nous pourrions commander avec les touches fléchées du clavier (FCTN S code 8, et FCTN D code 9) désignera la colonne dans laquelle il sera possible d'intervenir.

Nous créons donc le caractère 132 (ligne 190) qui est de couleur blanche (ligne 120).

Pour comprendre les calculs, il faut se souvenir de deux choses :

- Une relation entre deux grandeurs prend la valeur −1 lorsqu'elle est vraie et 0 (zéro) lorsqu'elle est fausse.
- Lors d'un test IF... THEN... , la machine teste la nullité de l'expression écrite entre IF et THEN. Si c'est nul, elle passe à la ligne suivante (ou au renvoi indiqué après ELSE), sinon elle saute au renvoi indiqué après le THEN.

Le sous-programme commençant à la ligne 320 efface donc la flèche située en XF, calcule la nouvelle valeur de XF suivant la touche pressée, et redessine la flèche au bon endroit.
Souvenons-nous que XF contiendra le numéro de la colonne où se trouve la flèche.

Comme pour le programme précédent, soyez exigeant et analysez toutes les déficiences de celui-ci.

```
10 REM ******************
15 REM * PROGRAMME No 3 *
20 REM *    TABLEAU     *
25 REM ******************
30 REM *    BASIC TI    *
35 REM ******************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM ******************
60 REM
100 CALL CLEAR
110 CALL SCREEN(6)
120 CALL COLOR(13,16,1)
130 DIM A(20),B(20),ART$(10),PU$
(10),QU$(10),PR$(10)
140 ART$(0)="ARTICLE"
150 PU$(0)="P.U"
160 QU$(0)="QTE."
170 PR$(0)="A PAYER"
180 CALL CHAR(123,"1010101010101
010")
190 CALL CHAR(132,"18181818FF7E3
C18")
200 T$=CHR$(123)
210 XF=7
220 PRINT TAB(2);ART$(0);TAB(11)
;PU$(0);TAB(16);QU$(0);TAB(21);P
R$(0)
230 FOR I=1 TO 21
240 PRINT TAB(1);T$;TAB(9);T$;TA
B(15);T$;TAB(20);T$;TAB(28);T$
250 NEXT I
260 GOSUB 320
270 CALL KEY(0,K,S)
280 IF S=0 THEN 270
290 IF (K<>9)*(K<>8)THEN 270
300 GOSUB 320
310 GOTO 270
320 CALL HCHAR(1,XF,32)
330 XF=XF-6*(K=9)+6*(K=8)
340 XF=-25*(XF>25)-7*(XF<7)+XF*(
XF>=7)*(XF<=25)
350 CALL HCHAR(1,XF,132)
360 RETURN
```

Programme 4 : Deux flèches

Notre client voudrait voir une seconde flèche qui indiquerait la ligne sur laquelle il pourrait écrire, et se déplacerait horizontalement conjointement avec la flèche du haut, et verticalement sous l'action des deux autres flèches du clavier (FCTN X code 10 et FCTN E code 11).

Cette seconde flèche est déterminée de la façon suivante :

- caractère 133 dessiné à la ligne 200;
- position horizontale : variable XF augmentée de 6. Il faut donc modifier la position des traits verticaux et créer entre eux un intervalle régulier de sept caractères;
- position verticale : la variable YF est calculée aux lignes 380 et 390 et est comprise entre 4 et 18. Elle augmente de 1 si K=10 et diminue de 1 si K=11.

La variable XF est calculée de telle façon que la flèche du haut désigne toujours le premier caractère de l'intitulé de la colonne.
Etudiez la ligne 310 où la multiplication permet de réaliser un "ET" logique. Il nous reste à écrire le contenu de la facture; c'est ce que nous allons entreprendre dans les programmes qui suivent.

```
10 REM ******************
15 REM * PROGRAMME No 4 *
20 REM *  DEUX FLECHES  *
25 REM ******************
30 REM *    BASIC TI    *
35 REM ******************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM ******************
60 REM
100 DIM A(20),B(20),ART$(10),PU$
(10),QU$(10),PR$(10)
110 CALL CLEAR
120 CALL COLOR(13,16,1)
130 CALL SCREEN(6)
140 ART$(0)=" ARTIC"
150 PU$(0)=" P.U."
160 QU$(0)=" QTE. "
170 PR$(0)=" PRIX"
180 CALL CHAR(123,"1010101010101
010")
190 CALL CHAR(132,"18181818FF7E3
C18")
200 CALL CHAR(133,"183878FFFF783
818")
210 T$=CHR$(123)
220 XF=4
230 YF=4
240 PRINT TAB(1);ART$(I);TAB(8);
PU$(I);TAB(15);QU$(I);TAB(22);PR
$(I)
250 FOR I=1 TO 21
260 PRINT TAB(1);T$;TAB(8);T$;TA
B(15);T$;TAB(22);T$
270 NEXT I
280 GOSUB 340
290 CALL KEY(0,K,S)
300 IF S=0 THEN 290
310 IF (K<>9)*(K<>8)*(K<>11)*(K<
>10)THEN 290
320 GOSUB 340
330 GOTO 290
340 CALL HCHAR(1,XF,32)
350 CALL HCHAR(YF,XF+6,123)
360 XF=XF-7*(K=9)+7*(K=8)
370 XF=-25*(XF>25)-4*(XF<4)+XF*(
XF>=4)*(XF<=25)
380 YF=YF-(K=10)+(K=11)
390 YF=-18*(YF>18)-4*(YF<4)+YF*(
YF>=4)*(YF<=18)
400 CALL HCHAR(1,XF,132)
410 CALL HCHAR(YF,XF+6,133)
420 RETURN
```

Programme 5 : Facture

Deux modifications apparaissent dans ce programme :

- l'introduction des données par l'utilisateur;
- le dessin des traits à l'aide de CALL VCHAR.

En ce qui concerne ce second point, il faut simplement remarquer que CALL VCHAR (comme CALL HCHAR) utilise 32 caractères par lignes, alors que TAB n'en utilise que 28. Le caractère 1 pour TAB devient donc le caractère 3 pour CALL VCHAR, et il devient possible de

dessiner le trait vertical à droite de la facture. Il faut d'autre part dessiner les traits après l'écriture du texte pour éviter l'effet de "scroll" (déroulement) qui ferait monter les lignes.

```
10 REM *****************
15 REM * PROGRAMME No 5 *
20 REM *     FACTURE    *
25 REM *****************
30 REM *    BASIC TI    *
35 REM *****************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM *****************
60 REM
100 DIM ART$(10),PU$(10),QU$(10)
,PR$(10)
110 CALL CLEAR
120 CALL COLOR(13,16,1)
130 CALL SCREEN(6)
140 ART$(0)=" ARTIC"
150 PU$(0)=" P.U."
160 QU$(0)=" QTE. "
170 PR$(0)=" PRIX"
180 CALL CHAR(123,"1010101010101
010")
190 CALL CHAR(132,"18181818FF7E3
C18")
200 CALL CHAR(133,"183878FFFF783
818")
210 T$=CHR$(123)
220 INPUT "NOMBRE D'ARTICLES : "
:NB
230 FOR I=1 TO NB
240 CALL CLEAR
250 PRINT TAB(9);"ARTICLE No ";I
260 PRINT : : :"DESIGNATION--> "
;
270 INPUT ART$(I)
280 INPUT "PRIX UNITAIRE -->":PU
$(I)
290 INPUT "QUANTITE --->":QU$(I)
300 PRIX=VAL(PU$(I))*VAL(QU$(I))
310 PR$(I)=STR$(PRIX)
320 NEXT I
330 CALL CLEAR
340 XF=4
350 YF=4
360 FOR I=0 TO NB
370 PRINT TAB(1);ART$(I);TAB(8);
PU$(I);TAB(15);QU$(I);TAB(22);PR
$(I)
380 NEXT I
390 FOR I=1 TO 20-NB
400 PRINT
410 NEXT I
420 FOR I=3 TO 32 STEP 7
430 CALL VCHAR(5,I,123,20)
440 NEXT I
450 GOSUB 510
460 CALL KEY(0,K,S)
470 IF S=0 THEN 460
480 IF (K<>9)*(K<>8)*(K<>11)*(K<
>10)THEN 460
490 GOSUB 510
500 GOTO 460
510 CALL HCHAR(1,XF,32)
520 CALL HCHAR(YF,XF+6,123)
530 XF=XF-7*(K=9)+7*(K=8)
540 XF=-25*(XF>25)-4*(XF<4)+XF*(
XF>=4)*(XF<=25)
550 YF=YF-(K=10)+(K=11)
560 YF=-18*(YF>18)-4*(YF<4)+YF*(
YF>=4)*(YF<=18)
570 CALL HCHAR(1,XF,132)
580 CALL HCHAR(YF,XF+6,133)
590 RETURN
600 L=LEN(MOT$)
610 FOR I=0 TO L-1
620 K=ASC(SEG$(MOT$,L-I,1))
630 CALL HCHAR(YF,XF+5-I,K)
640 NEXT I
650 RETURN
```

Programme 6 : Textes

Les traits dessinés par le programme précédent effacent le premier caractère de la première colonne; d'autre part, l'instruction PRINT provoquant un "scroll" de l'écran vers le haut, il sera difficile à l'utilisateur d'introduire un nouveau texte sans détruire le tableau.
Nous allons donc apporter deux améliorations :

- mieux structurer les variables utilisées pour le texte;
- créer une routine qui affiche un texte (de 6 caractères) en un point quelconque de l'écran.

Pour bien suivre le fonctionnement du programme (et pour bien suivre les exigences de l'utilisateur) nous avons représenté l'écran dans un rectangle de 24 lignes par 32 colonnes. Il est d'ailleurs conseillé d'avoir continuellement à sa disposition de tels modèles rectangulaires.

Nous allons remplacer les tableaux à un indice des programmes précédents par un tableau unique ART$(,) à deux indices.
Le premier indice est réservé au numéro de la colonne et prend donc les valeurs de 0 à 3.
Ainsi ART$(I,J) contiendra le texte qui doit être écrit à la Ième ligne de la colonne J. Attention, il s'agit ici de lignes et de colonnes de texte et non de celles de l'écran. Il faut donc trouver une relation entre I et J d'une part, et la position X,Y du texte sur l'écran.
Nous avons choisi de placer les intitulés des colonnes sur la deuxième ligne d'écran et la première ligne de la facture sur la sixième ligne d'écran. Un raisonnement simple permet alors de comprendre le but des lignes 380-390 et 480-490 du programme.

La routine commençant à la ligne 680 écrit un mot de six lettres nommé MOT$ en un point quelconque de l'écran : ligne X, colonne Y. Si le mot est formé de moins de six lettres, il est complété par des blancs : à droite pour le texte, à gauche pour les nombres (ce qui explique les lignes 690-720).

```
10 REM *****************
15 REM * PROGRAMME No 6 *
20 REM *     TEXTES     *
25 REM *****************
30 REM *    BASIC TI    *
35 REM *****************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM *****************
60 REM
100 DIM ART$(10,4)
110 CALL CLEAR
120 CALL COLOR(13,16,1)
130 CALL SCREEN(6)
140 ART$(0,0)=" ARTIC"
150 ART$(0,1)=" P.U. "
160 ART$(0,2)=" QTE. "
170 ART$(0,3)=" PRIX "
180 CALL CHAR(123,"1010101010101
010")
190 CALL CHAR(132,"18181818FF7E3
C18")
200 CALL CHAR(133,"183878FFFF783
818")
210 BL$="      "
220 INPUT "NOMBRE D'ARTICLES : "
:NB
230 FOR I=1 TO NB
240 CALL CLEAR
250 PRINT TAB(9);"ARTICLE ";I
260 PRINT : : :
270 INPUT "DESIGNATION --> ":ART
$(I,0)
280 INPUT "PRIX UNITAIRE -->":AR
T$(I,1)
290 INPUT "QUANTITE -->":ART$(I,
2)
300 PRIX=VAL(ART$(I,1))*VAL(ART$
(I,2))
310 ART$(I,3)=STR$(PRIX)
320 NEXT I
330 CALL CLEAR
340 XF=4
350 YF=4
360 FOR J=0 TO 3
```

```
370 MOT$=ART$(0,J)
380 X=J*7+3
390 Y=2
400 GOSUB 680
410 NEXT J
420 FOR I=3 TO 32 STEP 7
430 CALL VCHAR(3,I,123,20)
440 NEXT I
450 FOR I=1 TO NB
460 FOR J=0 TO 3
470 MOT$=ART$(I,J)
480 X=7*J+3
490 Y=5+I
500 GOSUB 680
510 NEXT J
520 NEXT I
530 GOSUB 590
540 CALL KEY(0,K,S)
550 IF S=0 THEN 540
560 IF (K<>9)*(K<>8)*(K<>11)*(K<
>10)THEN 540
570 GOSUB 590
580 GOTO 540
590 CALL HCHAR(1,XF,32)
600 CALL HCHAR(YF,XF+6,123)
610 XF=XF-7*(K=9)+7*(K=8)
620 XF=-25*(XF>25)-4*(XF<4)+XF*(
XF>=4)*(XF<=25)
630 YF=YF-(K=10)+(K=11)
640 YF=-18*(YF>18)-4*(YF<4)+YF*(
YF>=4)*(YF<=18)
650 CALL HCHAR(1,XF,132)
660 CALL HCHAR(YF,XF+6,133)
670 RETURN
680 L=LEN(MOT$)
690 IF J<>0 THEN 720
700 MOT$=MOT$&SEG$(BL$,1,6-L)
710 GOTO 730
720 MOT$=SEG$(BL$,1,6-L)&MOT$
730 FOR U=1 TO 6
740 K=ASC(SEG$(MOT$,U,1))
750 CALL HCHAR(Y,X+U,K)
760 NEXT U
770 RETURN
```

Programme 7 : Total

Nous ajoutons dans ce programme une routine qui calcule le total et l'affiche à la ligne 20, colonne 24. Ce programme devrait être aisément compris et nous n'insistons pas. Etudiez-le calmement...

```
10 REM ******************
15 REM * PROGRAMME No 7 *
20 REM *     TOTAL      *
25 REM ******************
30 REM *    BASIC TI    *
35 REM ******************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM ******************
60 REM
100 DIM ART$(10,4)
110 CALL CLEAR
120 CALL COLOR(13,16,1)
130 CALL SCREEN(6)
140 ART$(0,0)=" ARTIC"
150 ART$(0,1)=" P.U. "
160 ART$(0,2)=" QTE. "
170 ART$(0,3)=" PRIX "
180 CALL CHAR(123,"1010101010101
010")
190 CALL CHAR(132,"18181818FF7E3
C18")
200 CALL CHAR(133,"183878FFFF783
818")
210 BL$="      "
220 INPUT "NOMBRE D'ARTICLES : "
:NB
230 FOR I=1 TO NB
240 PRINT : : :
250 PRINT TAB(9);"ARTICLE ";I
260 INPUT "DESIGNATION --> ":ART
$(I,0)
270 INPUT "PRIX UNITAIRE -->":AR
T$(I,1)
280 INPUT "QUANTITE -->":ART$(I,
2)
290 PRIX=VAL(ART$(I,1))*VAL(ART$
(I,2))
300 ART$(I,3)=STR$(PRIX)
310 NEXT I
320 CALL CLEAR
330 XF=4
340 YF=4
350 FOR J=0 TO 3
360 MOT$=ART$(0,J)
370 X=J*7+3
380 Y=2
390 GOSUB 680
400 NEXT J
410 FOR I=3 TO 32 STEP 7
420 CALL VCHAR(3,I,123,20)
430 NEXT I
440 FOR I=1 TO NB
450 FOR J=0 TO 3
460 MOT$=ART$(I,J)
470 X=7*J+3
480 Y=5+I
490 GOSUB 680
500 NEXT J
510 NEXT I
520 GOSUB 780
530 GOSUB 590
540 CALL KEY(0,K,S)
550 IF S=0 THEN 540
560 IF (K<>9)*(K<>8)*(K<>11)*(K<
>10)THEN 540
570 GOSUB 590
580 GOTO 540
590 CALL HCHAR(1,XF,32)
600 CALL HCHAR(YF,XF+6,123)
610 XF=XF-7*(K=9)+7*(K=8)
620 XF=-25*(XF>25)-4*(XF<4)+XF*(
XF>=4)*(XF<=25)
630 YF=YF-(K=10)+(K=11)
640 YF=-18*(YF>18)-4*(YF<4)+YF*(
YF>=4)*(YF<=18)
650 CALL HCHAR(1,XF,132)
660 CALL HCHAR(YF,XF+6,133)
670 RETURN
680 L=LEN(MOT$)
690 IF J<>0 THEN 720
700 MOT$=MOT$&SEG$(BL$,1,6-L)
710 GOTO 730
720 MOT$=SEG$(BL$,1,6-L)&MOT$
730 FOR U=1 TO 6
740 K=ASC(SEG$(MOT$,U,1))
750 CALL HCHAR(Y,X+U,K)
760 NEXT U
770 RETURN
780 FOR T=1 TO NB
790 TOTAL=TOTAL+VAL(ART$(T,3))
800 NEXT T
810 TOT$=STR$(TOTAL)
820 X=24
830 Y=20
840 MOT$=TOT$
850 J=1
860 GOTO 680
```

Programme 8 : Variations

L'utilisateur veut pouvoir modifier sa facture, changer les prix ou les quantités, ajouter des lignes : nous lui offrons la possibilité de le faire en actionnant FCTN 8 (REDO) dont le code est 6. Ceci explique les modifications de la ligne de test 570, et la ligne 580 qui renvoie à la ligne 900 si REDO est actionnée.
Il faut maintenant calculer les indices I et J dans le tableau ART$(,) en fonction de la position XF et YF des deux flèches (ce qui est fait simplement aux lignes 900 et 910) et la position du nouveau texte (lignes 920 et 930).

Si l'utilisateur veut ajouter un article, le nombre contenu dans la variable NB doit augmenter : 950-970.
On efface les anciens contenus : lignes 980-1000.
Nouvelle scrutation du clavier : lignes 1010-1020.

Lorsque l'utilisateur a terminé son nouveau texte, il doit presser la touche "ENTER" (code 13). Ceci lui est permis grâce à la ligne 1030 qui renvoie à 1080. La suite du sous-pro-

gramme est évidente. Remarquez seulement la réinitialisation de la variable TOTAL (qui doit être évidemment recalculée avec les nouvelles données).

```
10 REM *****************
15 REM * PROGRAMME No 8 *
20 REM *   VARIATIONS   *
25 REM *****************
30 REM *    BASIC TI    *
35 REM *****************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM *****************
60 REM
100 DIM ART$(10,4)
110 CALL CLEAR
120 CALL COLOR(13,16,1)
130 CALL SCREEN(6)
140 ART$(0,0)=" ARTIC"
150 ART$(0,1)=" P.U. "
160 ART$(0,2)=" QTE. "
170 ART$(0,3)=" PRIX "
180 CALL CHAR(123,"1010101010101
010")
190 CALL CHAR(132,"18181818FF7E3
C18")
200 CALL CHAR(133,"183878FFFF783
818")
210 BL$="      "
220 INPUT "NOMBRE D'ARTICLES : "
:NB
230 FOR I=1 TO NB
240 PRINT : : :
250 PRINT TAB(9);"ARTICLE ";I
260 INPUT "DESIGNATION --> ":ART
$(I,0)
270 INPUT "PRIX UNITAIRE -->":AR
T$(I,1)
280 INPUT "QUANTITE -->":ART$(I,
2)
290 PRIX=VAL(ART$(I,1))*VAL(ART$
(I,2))
300 ART$(I,3)=STR$(PRIX)
310 NEXT I
320 CALL CLEAR
330 XF=4
340 YF=4
350 FOR J=0 TO 3
360 MOT$=ART$(0,J)
370 X=J*7+3
380 Y=2
390 GOSUB 710
400 NEXT J
410 FOR I=3 TO 32 STEP 7
420 CALL VCHAR(3,I,123,20)
430 NEXT I
440 FOR I=1 TO NB
450 FOR J=0 TO 3
460 MOT$=ART$(I,J)
470 X=7*J+3
480 Y=5+I
490 GOSUB 710
500 NEXT J
510 NEXT I
520 GOSUB 810
530 GOSUB 620
540 CALL SOUND(200,500,3,-4,5)
550 CALL KEY(0,K,S)
560 IF S=0 THEN 550
570 IF (K<>9)*(K<>8)*(K<>11)*(K<
>10)*(K<>6)THEN 540
580 IF K=6 THEN 900
590 GOSUB 620
600 GOTO 540
620 CALL HCHAR(1,XF,32)
630 CALL HCHAR(YF,XF+6,123)
640 XF=XF-7*(K=9)+7*(K=8)
650 XF=-25*(XF>25)-4*(XF<4)+XF*(
XF>=4)*(XF<=25)
660 YF=YF-(K=10)+(K=11)
670 YF=-18*(YF>18)-4*(YF<4)+YF*(
YF>=4)*(YF<=18)
680 CALL HCHAR(1,XF,132)
690 CALL HCHAR(YF,XF+6,133)
700 RETURN
710 L=LEN(MOT$)
720 IF J<>0 THEN 750
730 MOT$=MOT$&SEG$(BL$,1,6-L)
740 GOTO 760
750 MOT$=SEG$(BL$,1,6-L)&MOT$
760 FOR U=1 TO 6
770 K=ASC(SEG$(MOT$,U,1))
780 CALL HCHAR(Y,X+U,K)
790 NEXT U
800 RETURN
810 FOR T=1 TO NB
820 TOTAL=TOTAL+VAL(ART$(T,3))
830 NEXT T
840 TOT$=STR$(TOTAL)
850 X=24
860 Y=20
870 MOT$=TOT$
880 J=1
890 GOTO 710
900 J=(XF-3)/7
910 I=YF-5
920 X=XF-1
930 Y=YF
940 IF I<=NB THEN 980
950 NB=I
960 ART$(I,1)="0"
970 ART$(I,2)="0"
980 MOT$=""
990 ART$(I,J)=""
1000 GOSUB 710
1010 CALL KEY(0,C,S)
1020 IF S=0 THEN 1010
1030 IF C=13 THEN 1080
1040 ART$(I,J)=ART$(I,J)&CHR$(C)
1050 MOT$=ART$(I,J)
1060 GOSUB 710
1070 GOTO 1010
1080 PRIX=VAL(ART$(I,1))*VAL(ART
$(I,2))
1090 ART$(I,3)=STR$(PRIX)
1100 TOTAL=0
1110 GOTO 440
```

Programme 9 : Prix TTC

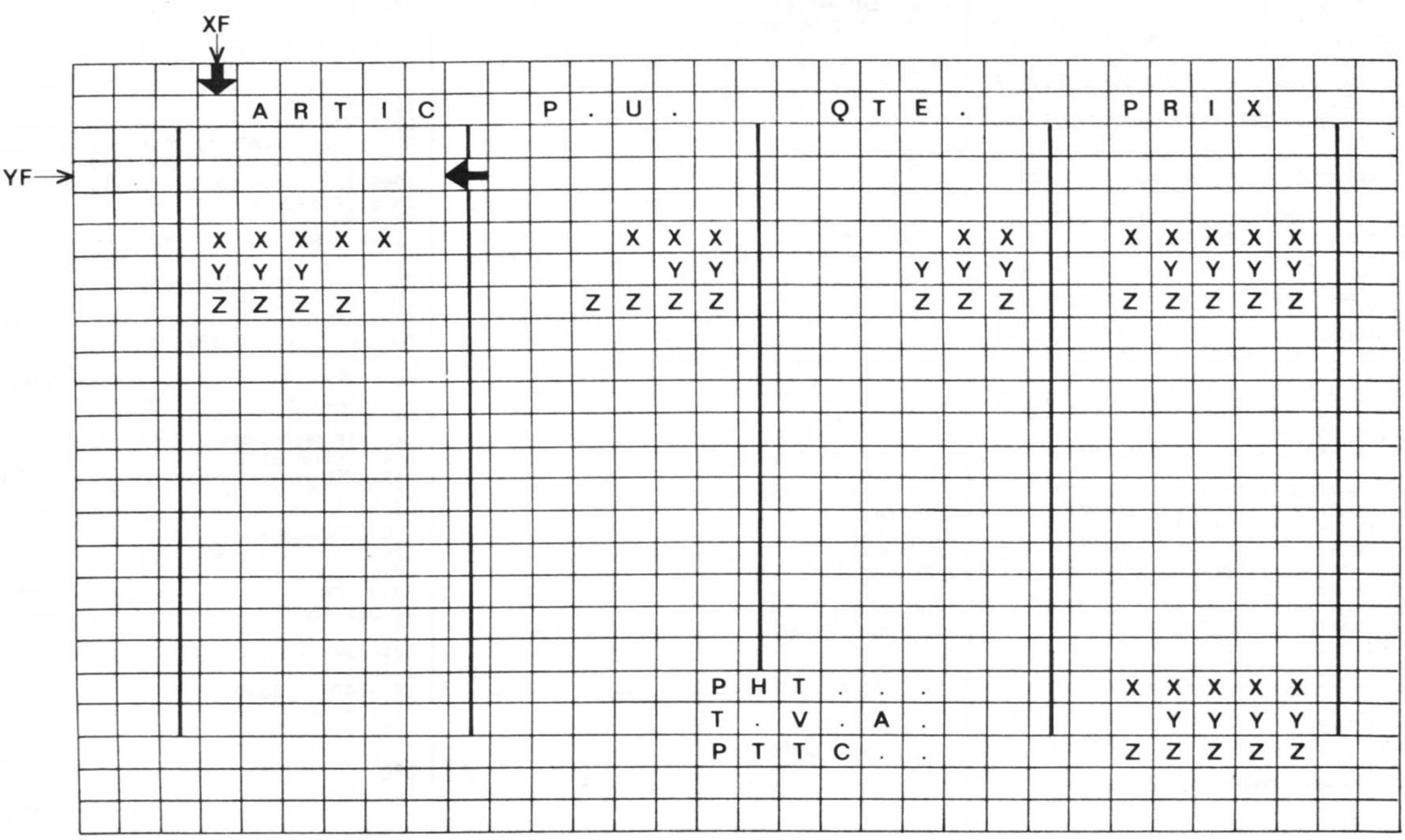
Nous laissons le soin au lecteur d'étudier ce programme qui apporte quelques modifications :

- quelques bips de tonalités différentes permettent d'éveiller l'attention de l'utilisateur ou de lui annoncer la fin d'une correction;
- la TVA et le prix TTC sont maintenant affichés et corrigés selon ses souhaits.

Ces programmes auraient besoin d'être encore améliorés. Si vous souhaitez nous proposer des modifications ou si vous désirez que nous vous en proposions, n'hésitez pas à nous écrire. ■

```
10 REM *****************
15 REM * PROGRAMME No 9 *
20 REM *    PRIX TTC    *
25 REM *****************
30 REM *    BASIC TI    *
35 REM *****************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM *****************
60 REM
100 DIM ART$(10,4)
110 CALL CLEAR
120 CALL COLOR(13,16,1)
130 CALL SCREEN(6)
140 ART$(0,0)=" ARTIC"
150 ART$(0,1)=" P.U. "
160 ART$(0,2)=" QTE. "
170 ART$(0,3)=" PRIX "
180 CALL CHAR(123,"1010101010101
010")
190 CALL CHAR(132,"18181818FF7E3
C18")
200 CALL CHAR(133,"183878FFFF783
818")
210 BL$="      "
220 INPUT "NOMBRE D'ARTICLES : "
:NB
230 FOR I=1 TO NB
240 PRINT : : :
250 PRINT TAB(9);"ARTICLE ";I
260 INPUT "DESIGNATION --> ":ART
$(I,0)
270 INPUT "PRIX UNITAIRE -->":AR
T$(I,1)
280 INPUT "QUANTITE -->":ART$(I,
2)
290 PRIX=VAL(ART$(I,1))*VAL(ART$
(I,2))
300 ART$(I,3)=STR$(PRIX)
```

```
310 CALL CLEAR
320 NEXT I
330 CALL CLEAR
340 CALL SOUND(300,120,2)
350 INPUT "TAUX DE TVA --> ":TVA
$
360 CALL CLEAR
370 XF=4
380 YF=4
390 FOR J=0 TO 3
400 MOT$=ART$(0,J)
410 X=J*7+3
420 Y=2
430 GOSUB 860
440 NEXT J
450 FOR I=3 TO 32 STEP 7
460 CALL VCHAR(3,I,123,20)
470 NEXT I
480 MOT$="PHT..."
490 X=15
500 Y=20
510 J=0
520 GOSUB 860
530 MOT$="T.V.A."
540 Y=21
550 GOSUB 860
560 MOT$="PTTC.."
570 Y=22
580 GOSUB 860
590 FOR I=1 TO NB
600 FOR J=0 TO 3
610 MOT$=ART$(I,J)
620 X=7*J+3
630 Y=5+I
640 GOSUB 860
650 NEXT J
660 NEXT I
670 GOSUB 960
680 GOSUB 770
690 CALL SOUND(200,500,3,-4,5)
700 CALL KEY(0,K,S)
710 IF S=0 THEN 700
720 IF (K<>9)*(K<>8)*(K<>11)*(K<
>10)*(K<>6)THEN 690
730 IF K=6 THEN 1130
740 GOSUB 770
750 GOTO 690
770 CALL HCHAR(1,XF,32)
780 CALL HCHAR(YF,XF+6,123)
790 XF=XF-7*(K=9)+7*(K=8)
800 XF=-25*(XF>25)-4*(XF<4)+XF*(
XF)=4)*(XF<=25)
810 YF=YF-(K=10)+(K=11)
820 YF=-18*(YF>18)-4*(YF<4)+YF*(
YF)=4)*(YF<=18)
830 CALL HCHAR(1,XF,132)
840 CALL HCHAR(YF,XF+6,133)
850 RETURN
860 L=LEN(MOT$)
870 IF J<>0 THEN 900
880 MOT$=MOT$&SEG$(BL$,1,6-L)
890 GOTO 910
900 MOT$=SEG$(BL$,1,6-L)&MOT$
910 FOR U=1 TO 6
920 K=ASC(SEG$(MOT$,U,1))
930 CALL HCHAR(Y,X+U,K)
940 NEXT U
950 RETURN
960 FOR T=1 TO NB
970 TOTAL=TOTAL+VAL(ART$(T,3))
980 NEXT T
990 TOT$=STR$(TOTAL)
1000 X=24
1010 Y=20
1020 MOT$=TOT$
1030 J=1
1040 GOSUB 860
1050 TAXE=TOTAL*VAL(TVA$)/100
1060 MOT$=SEG$(STR$(TAXE),1,6)
1070 Y=21
1080 GOSUB 860
1090 PTTC=TOTAL+TAXE
1100 MOT$=SEG$(STR$(PTTC),1,6)
1110 Y=22
1120 GOTO 860
1130 J=(XF-3)/7
1140 I=YF-5
1150 X=XF-1
1160 Y=YF
1170 IF I<=NB THEN 1210
1180 NB=I
1190 ART$(I,1)="0"
1200 ART$(I,2)="0"
1210 MOT$=""
1220 ART$(I,J)=""
1230 GOSUB 860
1240 CALL KEY(0,C,S)
1250 IF S=0 THEN 1240
1260 IF C=13 THEN 1310
1270 ART$(I,J)=ART$(I,J)&CHR$(C)
1280 MOT$=ART$(I,J)
1290 GOSUB 860
1300 GOTO 1240
1310 PRIX=VAL(ART$(I,1))*VAL(ART
$(I,2))
1320 ART$(I,3)=STR$(PRIX)
1330 TOTAL=0
1340 GOTO 590
```



Jeu de mémorisation

Jean-Claude Cornevin

Le principe du jeu est simple : 18 paires de cartes sont retournées. Chaque joueur, à tour de rôle, retourne deux cartes.
Si les cartes sont identiques, elles sont retirées du jeu, le score du joueur est augmenté d'un point.
Si les cartes sont différentes, elles sont à nouveau retournées et le tour passe au joueur suivant.

Vous pourrez, avec ce jeu écrit en **Basic TI**, tester et exercer votre mémoire visuelle.

Les remarques insérées dans le listing vous permettront de comprendre le fonctionnement de ce programme. Que le meilleur gagne... ■




```
100 REM *******************
110 REM *      JEU DE      *
120 REM *  MEMORISATION    *
130 REM *******************
140 REM *    BASIC  TI     *
150 REM *******************
160 REM *    COPYRIGHT     *
170 REM *  J.C. CORNEVIN   *
180 REM * ET  99 MAGAZINE *
190 REM *******************
200 CALL CLEAR
210 CALL SCREEN(2)
220 PRINT "                 JEU": : : : : :
230 PRINT "                  DE": : : : : :
240 PRINT "        MEMORISATION": : : :
250 FOR I=2 TO 9
260 CALL COLOR(I,16,1)
270 NEXT I
280 CALL COLOR(14,10,16)
290 GOSUB 3740
300 REM------------------------------ MOTI
F CARTES ----------------------------
---
310 FOR I=90 TO 156
320 READ A$
330 CALL CHAR(I,A$)
340 NEXT I
350 DATA 3F7F7F7F,000000000010101,000000
0000808080,0101010307070707,808080C0E0E
0E0E0,0F0701
360 DATA FFFFFFFFFDFDF9F9,FFFFFF7F3F3F1F0
F,F1F1E1E1C1C181FF,0703030101000FF,00C
0F0FFFFFFFFFF
370 DATA 000001FFFFFFFFFF,FFFF3F0F010081F
3,F0E0C0F0FC3F7FFF,000000003F4F8787,000
00000FCF2E1E1
380 DATA 8787474F3FBF3F3F,E1E1E2F2FCFCFCF
C,3F3F1F1F0F03,FCFCF8F8F0C0,
390 DATA ,8855225588552255,000000000001C3E
7F,000000000387CFE,7F7F7F3F1F0F0703
400 DATA FEFEFEFCF8F0E0C0,01,8,0018187E7E
1818,0000000000000001,000000081020408,
030303070F1F1F
410 DATA C0E0E0E0E0E0E0F,3F3F3F3F3F1C,F
8F8F8F8F8F0C,,,000000151F1F1F0F
420 DATA 00000054FCFCFCF8,0706060607070
7,F0F0F0F0F0F0707,070707070707,70F0F0F0
F,0100000103
430 DATA F860F0F8FC,00000000001F3F7F,7F7F
7F7F3F3F1F,E0F0F0F0F0E0C
440 DATA 00000000FCFCFCFC,,0000001818,000
0000101,000000808,00000107070F0F07
450 DATA 000080E0E0F0F0E,1B7C7FFFFF7F7D1B
,D8BE7EFFFFFEBED8,0307070F1F,C0E0E0F0F8
,,
460 DATA 00001F003FBF3F,00000606FEFEFE,0C
0C,FF087F53577F001,8000FFFCF0C
470 FOR I=33 TO 46
480 READ A$
490 CALL CHAR(I,A$)
500 NEXT I
510 DATA "0000202030F0D0F","000040C0C0C0
E0E0","1F3F7C1831010303","F0F0F8F8F8F8F
8F8","00","00","00"
520 DATA "0000000000616325","000000000086
C6A4","27371F0F0F2F373B","E4ECF8F0F0F4E
CDC","3D1F0E047C70"
530 DATA "BCF870203C0C","0000060810204080
"
540 CALL CHAR(58,"FFFFFFFFFFF8FFFF")
550 CALL CHAR(59,"FFFFFFFF9F4F0F9F")
560 CALL CHAR(60,"F8F1F2E4E4E2F1F0")
570 CALL CHAR(61,"7F3F3F1F1F1F0F8F")
580 CALL CHAR(62,"FCFAF6CEE9FDFFFF")
590 CALL CHAR(63,"4F2783E0FFFFFFFF")
600 CALL CHAR(64,"F0E080")
610 CALL CHAR(87,"0078787878787878")
620 DIM A(18)
630 REM-------------------------------PRES
ENTATION-----------------------------
---
640 CALL CLEAR
650 PRINT "POUR CONSULTER LES REGLES   1":
 : : : :
660 PRINT "POUR COMMENCER LE JEU      2":
 : : :
670 CALL KEY(0,K,S)
680 IF S<>1 THEN 670
690 IF K=49 THEN 3650
700 CALL CLEAR
710 JU=0
720 JO=0
730 JT=0
740 PRINT "NOM DU 1ER JOUEUR": : :
750 INPUT J1$
760 FOR I=1 TO 10
770 CALL GCHAR(23,I+4,PLAY(1,I))
780 NEXT I
790 CALL CLEAR
800 PRINT "NOM DU 2EME JOUEUR": : :
810 INPUT J2$
820 FOR I=1 TO 10
830 CALL GCHAR(23,I+4,PLAY(2,I))
840 NEXT I
850 GOSUB 3190
860 CALL CLEAR
870 CALL CHAR(81,"78F8F8F8F8")
880 REM----------------------------- MISE
 EN COULEUR--------------------------
---
890 CALL COLOR(1,2,1)
900 CALL COLOR(2,4,16)
910 FOR I=3 TO 8
920 CALL COLOR(I,2,16)
930 NEXT I
940 FOR I=1 TO 5
950 CALL COLOR(I+8,4+2*I,16)
960 NEXT I
970 CALL COLOR(14,10,16)
980 CALL COLOR(15,4,16)
990 CALL COLOR(16,2,6)
1000 CALL SCREEN(3)
1010 REM---------------------------------
CODAGE ------------------------------
----
1020 PRINT "                ooJOUEooXooooYooo"
1030 FOR I=1 TO 18
1040 A(I)=2
1050 NEXT I
1060 FOR Y=1 TO 4
1070 FOR X=1 TO 9
1080 RANDOMIZE
1090 Z=INT(RND*17)+1
1100 IF A(Z)<1 THEN 1140
1110 A(Z)=A(Z)-1
1120 ME(X,Y)=Z
1130 GOTO 1180
1140 Z=Z+1
1150 IF Z<19 THEN 1100
1160 Z=1
1170 GOTO 1100
1180 GOSUB 3330
1190 NEXT X
1200 NEXT Y
1210 REM------------------------ MISE EN
 PLACE TEXTE ------------------------
---
1220 CALL CHAR(159,"FFFFFFFFFFFFFFFF")
1230 CALL HCHAR(1,16,88)
1240 CALL VCHAR(12,2,89)
1250 FOR I=1 TO 9
1260 X=3+3*I
1270 C=48+I
1280 CALL HCHAR(3,X,C)
1290 IF I>4 THEN 1320
1300 Y=2+4*I
1310 CALL HCHAR(Y,4,C)
1320 NEXT I
1330 JOU=1
1340 REM------------------------- AFFICH
AGE JOUEUR --------------------------
---
1350 FOR I=1 TO 10
1360 CALL HCHAR(23,I+2,PLAY(JOU,I))
1370 NEXT I
1380 REM---------------------------- DEBUT
 DU JEU -----------------------------
---
1390 XX(2)=0
1400 FOR I=1 TO 2
1410 CALL HCHAR(23,23,159)
1420 CALL HCHAR(23,28,159)
1430 CALL KEY(0,K,ST)
1440 IF ST<>1 THEN 1430
1450 IF K>57 THEN 1430
1460 IF K<49 THEN 1430
1470 X=K-48
1480 XX(I)=X
1490 CALL VCHAR(23,23,K)
1500 CALL KEY(0,K,ST)
1510 IF ST<>1 THEN 1500
1520 IF K>52 THEN 1500
1530 IF K<49 THEN 1500
1540 Y=K-48
1550 YY(I)=Y
1560 CALL VCHAR(23,28,K)
1570 IF ME(X,Y)=0 THEN 1410
1580 IF XX(1)<>XX(2)THEN 1600
1590 IF YY(1)=YY(2)THEN 1410
1600 MM(I)=ME(X,Y)
1610 ON MM(I)GOSUB 1930,2000,2070,2140,22
10,2280,2350,2420,2490,2560,2630,2700,2
770,2840,2910,2980,3050,3120
1620 GOSUB 3330
1630 NEXT I
1640 FOR I=1 TO 400
1650 NEXT I
1660 REM--------------------- COMPARAISON
 DES CARTES -------------------------
---
1670 IF MM(1)=MM(2)THEN 1820
1680 GOSUB 3190
1690 REM----------------------- CHANGEM.
DE JOUEUR ---------------------------
---
1700 JOU=JOU+1
1710 IF JOU<3 THEN 1740
1720 JOU=1
1730 REM--------------------- RETOURNEME
NT DES CARTES-----------------------
---
1740 X=XX(1)
1750 Y=YY(1)
1760 GOSUB 3330
1770 X=XX(2)
1780 Y=YY(2)
1790 GOSUB 3330
1800 GOTO 1340
1810 REM---------------------------- AUGM
. DU SCORE--------------------------
---
```

```
1820 IF JOU=1 THEN 1850
1830 JU=JU+1
1840 GOTO 1860
1850 JO=JO+1
1860 JT=JO+JU
1870 REM---------------------- EFFACEMEN
T DES CARTES--------------------------
---
1880 GOSUB 3260
1890 ME(XX(1),YY(1))=0
1900 ME(XX(2),YY(2))=0
1910 IF JT=18 THEN 3430
1920 GOTO 1740
1930 B=136
1940 C=46
1950 D=137
1960 E=138
1970 F=150
1980 G=150
1990 RETURN
2000 B=102
2010 C=103
2020 D=152
2030 E=153
2040 F=154
2050 G=103
2060 RETURN
2070 B=102
2080 C=103
2090 D=155
2100 E=156
2110 F=102
2120 G=103
2130 RETURN
2140 B=141
2150 C=141
2160 D=141
2170 E=141
2180 F=141
2190 G=141
2200 RETURN
2210 B=144
2220 C=145
2230 D=146
2240 E=147
2250 F=148
2260 G=149
2270 RETURN
2280 B=104
2290 C=105
2300 D=106
2310 E=107
2320 F=108
2330 G=109
2340 RETURN
2350 B=119
2360 C=119
2370 D=119
2380 E=119
2390 F=119
2400 G=119
2410 RETURN
2420 B=120
2430 C=46
2440 D=122
2450 E=123
2460 F=124
2470 G=125
2480 RETURN
2490 B=128
2500 C=129
2510 D=130
2520 E=131
2530 F=132
2540 G=133
2550 RETURN
2560 B=40
2570 C=41
2580 D=42
2590 E=43
2600 F=44
2610 G=45
2620 RETURN
2630 B=96
2640 C=97
2650 D=98
2660 E=99
2670 F=100
2680 G=101
2690 RETURN
2700 B=113
2710 C=114
2720 D=115
2730 E=116
2740 F=117
2750 G=118
2760 RETURN
2770 B=142
2780 C=143
2790 D=142
2800 E=143
2810 F=142
2820 G=143
2830 RETURN
2840 B=58
2850 C=59
2860 D=60
2870 E=61
2880 F=62
2890 G=63
2900 RETURN
2910 B=150
2920 C=139
2930 D=150
2940 E=87
2950 F=90
2960 G=81
2970 RETURN
2980 B=33
2990 C=34
3000 D=35
3010 E=36
3020 F=134
3030 G=135
3040 RETURN
3050 B=91
3060 C=92
3070 D=93
3080 E=94
3090 F=95
3100 G=64
3110 RETURN
3120 B=141
3130 C=141
3140 D=142
3150 E=143
3160 F=141
3170 G=141
3180 RETURN
3190 B=112
3200 C=112
3210 D=112
3220 E=112
3230 F=112
3240 G=112
3250 RETURN
3260 B=37
3270 C=37
3280 D=37
3290 E=37
3300 F=37
3310 G=37
3320 RETURN
3330 REM-----------------------AFFICHAGE
 CARTES -----------------------------
---
3340 X1=3+3*X
3350 Y1=1+4*Y
3360 CALL HCHAR(Y1,X1,B)
3370 CALL HCHAR(Y1,X1+1,C)
3380 CALL HCHAR(Y1+1,X1,D)
3390 CALL HCHAR(Y1+1,X1+1,E)
3400 CALL HCHAR(Y1+2,X1,F)
3410 CALL HCHAR(Y1+2,X1+1,G)
3420 RETURN
3430 CALL CLEAR
3440 REM------------------------- AFFICHA
GE SCORE ----------------------------
---
3450 CALL SCREEN(16)
3460 IF JO<>JU THEN 3490
3470 PRINT "EGALITE 9 A 9": : :
3480 GOTO 3580
3490 PRINT "AU SCORE DE ";JO;" A ";JU: :
:
3500 PRINT "                 GAGNE": : :
3510 IF JO>JU THEN 3540
3520 Q=2
3530 GOTO 3550
3540 Q=1
3550 FOR I=1 TO 10
3560 CALL HCHAR(21,I+2,PLAY(Q,I))
3570 NEXT I
3580 GOSUB 3740
3590 PRINT "POUR REJOUER      1": :
3600 PRINT "POUR ARRETER      2": :
3610 INPUT R
3620 IF R=1 THEN 700
3630 END
3640 REM--------------------------- AFFICH.
REGLES --------------------------
---
3650 CALL CLEAR
3660 PRINT "                 REGLES": : :
3670 PRINT "IL S'AGIT D'UN JEU  FAISANT":
 :"APPEL A LA MEMOIRE": :"CHAQUE JOUEUR
 A TOUR DE ROLE": :
3680 PRINT "RETOURNE 2 CARTES": :"SI LES
DEUX IMAGES SONT IDEN": :"TIQUES  LE  J
OUEUR GAGNE  UN": :
3690 PRINT "POINT ET CONTINUE A JOUER": :
"SI LES IMAGES  SONT DIFFEREN": :"TES L
ES CARTES SE RETOURNENT": :
3700 PRINT "ET  LE  TOUR PASSE A L AUTRE"
: :"JOUEUR": :"APPUYER ENTER POUR CONTI
NUER"
3710 CALL KEY(0,K,S)
3720 IF S<>1 THEN 3710
3730 GOTO 700
3740 REM---------------------------SONORIS
ATION-----------------------------
---
3750 CALL SOUND(200,330,5)
3760 CALL SOUND(200,330,5)
3770 CALL SOUND(200,349,5)
3780 CALL SOUND(200,393,5)
3790 CALL SOUND(200,330,5)
3800 CALL SOUND(200,330,5)
3810 CALL SOUND(400,330,5)
3820 CALL SOUND(200,294,5)
3830 CALL SOUND(200,247,5)
3840 CALL SOUND(200,349,5)
3850 CALL SOUND(200,392,5)
3860 CALL SOUND(200,440,5)
3870 CALL SOUND(200,392,5)
3880 CALL SOUND(200,349,5)
3890 CALL SOUND(200,330,5)
3900 CALL SOUND(200,294,7)
3910 CALL SOUND(200,247,5)
3920 CALL SOUND(200,262,7)
3930 CALL SOUND(200,294,5)
3940 CALL SOUND(400,330,6)
3950 RETURN
```

Jack-Pot

Marianne Sutz

Ce programme, écrit en **Basic Etendu**, est très largement inspiré du célèbre jeu de casino du même nom. Dans un premier temps, le programme vous demande de rentrer la somme que vous désirez miser. La mise doit être inférieure à 10000.

Vous devez ensuite choisir les symboles à modifier. Par exemple, si vous frappez les touches 4 et 2 (l'ordre n'a pas d'importance), les symboles correspondant aux positions 2 et 4 seront changés. Si vous faites une erreur, vous pouvez tout annuler avec la touche 0. Une fois votre choix déterminé, vous devrez le valider avec la barre d'espacement.

La partie s'arrête lorsque vous obtenez quatre symboles identiques ou si votre score est inférieur ou égal à zéro. Avant de vous indiquer comment s'effectue le calcul des points en fonction des symboles visualisés, il faut signaler que l'apparition des flèches provoque un retirage aléatoire sur les quatre positions. Points obtenus (ou perdus) :

- quatre symboles quelconques identiques : +50 - fin de partie;
- quatre "dollars" : +100 - fin de partie;
- trois symboles quelconques identiques : +10;
- trois symboles quelconques identiques plus "dollar" : +20;
- trois "dollars" : +25;
- deux fois deux symboles quelconques identiques : score identique;
- deux symboles quelconques identiques : −5;
- couleurs (trèfle, coeur, carreau, pique) : +100;
- tous les symboles différents : −25%.

En fin de partie, le programme vous demande si vous désirez rejouer. Si vous répondez par la négative, une récapitulation des gains et pertes sera affichée. Une pression sur n'importe quelle touche interrompra définitivement le programme. Bonne chance... ■

```
100 ! **********************
110 ! *     Jack - Pot     *
120 ! **********************
130 ! *     Copyright      *
140 ! *   Marianne Sutz    *
150 ! * et  "99 Magazine"  *
160 ! **********************
170 ! *   Basic  Etendu    *
180 ! **********************
190 !
200 E$="3333F3F30303FFFF" :: F$="FFFFFFFF
FFFFFEFE" :: G$="FFFEFCF8F0E0C080" :: H
$="0103070F1F3F7FFF" :: Q$="FF7F3F1F0F070
301"
210 S$="FFFFC0C0CFCFCCCC" :: T$=RPT$("C",
16):: W$="FFFF0000FFFF0000" :: U$="CCCC
CFCFC0C0FFFF" :: Y$="0000FFFF0000FFFF" ::
 X$="FFFF0303F3F33333"
220 Z$=RPT$("3",16):: E$="3333F3F30303FFF
F"
230 R$="80C0E0F0F8FCFEFF" :: V$=RPT$("0",
16):: P$=RPT$("F",16):: OPTION BASE 1 :
: DIM GR$(10,4),C(10,2),SC(20,3)
240 C(1,1),C(1,2)=9 :: C(2,1),C(2,2),C(5,
2)=3 :: C(3,1),C(3,2)=14 :: C(4,1),C(4,
2),C(5,1)=10
```

```
250 C(6,1),C(6,2)=12 :: C(7,1),C(7,2)=13
:: C(8,1),C(8,2)=16 :: C(9,1),C(9,2)=2
:: C(10,1),C(10,2)=11
260 GR$(1,1)="070F1F3F3F7F7FFFFFFFFFFFFFFF
F7F7FC0F0F8FCFCFEFEFE"&P$ :: GR$(1,3)="
030F1F3F3F7F7F7F"&P$&"E0F0F8FCFCFEFEFF"&F
$
270 GR$(1,2)="3F3F1F1F0F070301010000000000
00000"&P$&"FFFF7F3F1F070301" :: GR$(1,4
)=P$&"FFFFFEFCF8E0C080FCFCF8F8F0E0C08080"
280 GR$(2,1)=V$&"00000F3F7F7FFFFF030F1F1F
3F3F3F3F1F1F0FCFFFFFFFFFF" :: GR$(2,2)="
FFFF7F7F3F0F0000"&V$&"FFFFFFEFC707070707070
70F0F0F1F1F3F"
290 GR$(2,3)="C0F0F8F8FCFCFCFCF8F8F0F3FFF
FFFFF"&V$&"0000F0FCFEFEFFFF" :: GR$(2,4
)="FFFFFFF7E3E0E0E0E0E0F0F0F0F8F8FCFFFFFE
FEFCF0000000"
300 GR$(3,1)=V$&"0000010307070F0F01010103
070F1F3F7FFFFFFFFFFFFFFF" :: GR$(3,2)="
1F1F1F1F1F1F1F0F0F0701000000000000"&P$&"FFF
BE303030707 0F"
310 GR$(3,3)="808080C0E0F0F8FCFEFFFFFFFFFF
FFFFF"&V$&"000080C0E0E0F0F0" :: GR$(3,4
)=P$&"FFDFC7C0C0E0E0F0F8F8F8F8F8F8F0F0E
080"
320 GR$(4,2)=Q$&V$&P$&Q$ :: GR$(4,3)=R$&P
$&V$&R$ :: GR$(4,4)=P$&G$&G$&V$ :: GR$(
5,2)="FFFFFFFFFFFFF7F7F7F3F3F1F1F0F0703"&P
$&"FFFFFFFFFEFCF8F0"
330 GR$(5,1)="00000000000000001070F1F3F3F7
F7FFF000000000000000F8FCFEFFFFFFFFFFFFFF"
:: GR$(5,4)="C0E0E0E0E0C0C0C080800000000
0000"&V$&V$
340 GR$(5,3)="000000010307 0E0C1838306060C
0C080387CFEC6870303030 0" :: GR$(6,1)="0
00000000000103070F0F1F3F7FFFFF000000073F
FFFFFF"&P$
350 GR$(6,2)="FFFF7F3F1F0F0F0703010000000
00000"&P$&"FFFFFF3F07" :: GR$(6,3)="000
000E0FCFFFFFF"&P$&"00000000000080C0E0F0F0
F8FCFEFFFF"
360 GR$(6,4)=P$&"FFFFFFFCE0000000FFFFFEFC
F8F0F0E0C080" :: GR$(7,1)=V$&"00070F1F1
F1F0F7F0000000000001071FFFFFFFFFFFFFF"
370 GR$(7,2)="FF8F1F1F1F0F0700"&V$&"FFFFF
FFFFFFFFF1F0701" :: GR$(7,3)="0000071F3
FFFFFFF"&P$&"0000C0F8FEFEFFFF"&F$
380 GR$(7,4)=P$&"FFFFFF3F1F070000FEFEFFFF
FFFFFFFFFFFFFEFEF8C0" :: GR$(8,1)=S$&T$
&W$&S$ :: GR$(8,2)=T$&U$&U$&Y$ :: GR$(8,3
)=W$&X$&X$&Z$
390 GR$(8,4)=E$&Y$&Z$&E$ :: GR$(9,1)=V$&"
0002060E1F3760C00103060C183C7C0C0C0C0C0
CFCFC"
400 GR$(9,2)="C060371F0E060200"&V$&"0000F
CFC0C0C0C0C0C7C3C180C060301" :: GR$(9,3
)="80C06030183C3E303030303 03F3F0000"&V$&"
00406070F8EC0603"
410 GR$(9,4)="00003F3F30303030303E3C18306
0C0800306ECF87060400000"
420 GR$(10,1),GR$(10,3)="0202020F1F3F7A72
72727272727A3F1F404040F0F8FC5E4E4E40404
04040F0F8"
430 GR$(10,2),GR$(10,4)="0F020202020202 72
727A3F1F0F020202FC5E4E4E4E4E4E4E5EFCF
8F0404040" :: GR$(4,1)=V$&H$&H$&P$
440 CALL CLEAR :: CALL SCREEN(5):: CALL M
AGNIFY(3):: CALL COLOR(2,10,2):: B2=48
:: CALL VCHAR(1,9,42,5):: CALL HCHAR(24,4
,42,4)
450 FOR B=1 TO 3 :: CALL VCHAR(1,B,42,24)
:: CALL HCHAR(B,4,42,4):: NEXT B :: FOR
 B=8 TO 10 STEP 2 :: CALL VCHAR(1,B,42,24
):: NEXT B
460 FOR B=6 TO 21 STEP 5 :: CALL VCHAR(B+
2,9,42,4):: CALL HCHAR(B+2,4,42,4):: B2
=B2+1 :: CALL VCHAR(B,9,B2):: NEXT B
470 CALL CHAR(136,GR$(10,1),140,GR$(10,2)
,60,GR$(10,3),44,GR$(10,4))
480 CALL CHAR(88,GR$(8,1),92,GR$(8,2),96,
GR$(8,3),100,GR$(8,4))
490 CALL CHAR(104,GR$(2,1),108,GR$(2,2),1
12,GR$(2,3),116,GR$(2,4))
500 CALL CHAR(120,GR$(1,1),124,GR$(1,2),1
28,GR$(1,3),132,GR$(1,4))
510 CALL SPRITE(#1,136,11,25,25,#2,140,11
,41,25,#3,60,11,25,41,#4,44,11,41,41)
520 CALL SPRITE(#5,88,16,65,25,#6,92,16,8
1,25,#7,96,16,65,41,#8,100,16,81,41)
530 CALL SPRITE(#9,104,3,105,25,#10,108,3
,121,25,#11,112,3,105,41,#12,116,3,121,
41)
540 CALL SPRITE(#13,120,9,145,25,#14,124,
9,161,25,#15,128,9,145,41,#16,132,9,161
,41)
550 FOR B=1 TO 24 :: DISPLAY AT(B,11)BEEP
:"$$$ JACK POT $$$" :: NEXT B :: FOR D=
0 TO 200 :: NEXT D :: GOSUB 1300
560 ON WARNING NEXT :: DISPLAY AT(3,10):"
VOTRE MISE SVP" :: ACCEPT AT(3,25)SIZE(
4)VALIDATE(DIGIT)BEEP:MI :: DISPLAY AT(3,
10):""
570 DISPLAY AT(3,11):"MISE";MI;"FRANCS" :
: CALL HCHAR(5,13,42,15):: CALL HCHAR(1
1,13,42,15):: CALL VCHAR(6,13,42,5):: CAL
L VCHAR(6,27,42,5)
580 DISPLAY AT(7,15)SIZE(8):"CREDIT:" ::
DISPLAY AT(8,12)SIZE(12):MI;"FRANCS"
590 XX=XX+1 :: SC(XX,1),CR=MI :: DISPLAY
AT(20,10):"TAPER LES NUMEROS" :: DISPLA
Y AT(21,13):"DE DESSIN A" :: DISPLAY AT(2
2,13):"REMPLACER ET"
600 DISPLAY AT(23,14):"FRAPPER LA" :: DIS
PLAY AT(24,10):"BARRE D'ESPACEMENT" ::
X1=10 :: X2=8 :: X3=2 :: X4=1
610 DA,DB,DC,DD=0
620 CALL KEY(0,R1,ST):: IF R1=32 THEN 680
 ELSE IF R1=48 THEN GOSUB 1310 :: GOTO
610 ELSE IF R1<49 OR R1>52 OR ST=0 OR ST=
-1 THEN 620
630 IF R1=49 THEN GOSUB 1280 :: DA=1 :: C
ALL HCHAR(7,9,43)
640 IF R1=50 THEN GOSUB 1280 :: DB=2 :: C
ALL HCHAR(12,9,43)
650 IF R1=51 THEN GOSUB 1280 :: DC=4 :: C
ALL HCHAR(17,9,43)
660 IF R1=52 THEN GOSUB 1280 :: DD=8 :: C
ALL HCHAR(22,9,43)
670 GOTO 620
680 X=DA+DB+DC+DD :: IF X=0 THEN 620
690 FOR B=0 TO 30 :: CALL SOUND(50,440,B,
-4,B):: NEXT B :: ON X GOTO 700,730,700
,760,700,730,700,790,700,730,700,760,700,
730,700
700 GOSUB 1270 :: X1=CA :: CALL CHAR(136,
GR$(CA,1),140,GR$(CA,2),60,GR$(CA,3),44
,GR$(CA,4))
710 CALL PATTERN(#1,136,#2,140,#3,60,#4,4
4):: CALL COLOR(#1,C(CA,1),#2,C(CA,1),#
3,C(CA,2),#4,C(CA,1))
720 GOSUB 1290 :: IF X=5 OR X=13 THEN 760
 ELSE IF X=9 THEN 790 ELSE IF X=1 THEN
810
730 GOSUB 1270 :: X2=CA :: CALL CHAR(88,G
R$(CA,1),92,GR$(CA,2),96,GR$(CA,3),100,
GR$(CA,4))
740 CALL PATTERN(#5,88,#6,92,#7,96,#8,100
```

```
):: CALL COLOR(#5,C(CA,1),#6,C(CA,1),#7
,C(CA,2),#8,C(CA,1))
750 GOSUB 1290 :: IF X=10 OR X=11 THEN 79
0 ELSE IF X=2 OR X=3 THEN 810
760 GOSUB 1270 :: X3=CA :: CALL CHAR(104,
GR$(CA,1),108,GR$(CA,2),112,GR$(CA,3),1
16,GR$(CA,4))
770 CALL PATTERN(#9,104,#10,108,#11,112,#
12,116):: CALL COLOR(#9,C(CA,1),#10,C(C
A,1),#11,C(CA,2),#12,C(CA,1))
780 GOSUB 1290 :: IF X=4 OR X=5 OR X=6 OR
 X=7 THEN 810
790 GOSUB 1270 :: X4=CA :: CALL CHAR(120,
GR$(CA,1),124,GR$(CA,2),128,GR$(CA,3),1
32,GR$(CA,4))
800 CALL PATTERN(#13,120,#14,124,#15,128,
#16,132):: CALL COLOR(#13,C(CA,1),#14,C
(CA,1),#15,C(CA,2),#16,C(CA,1)):: GOSUB 1
290
810 GOSUB 1310 :: IF X1=X2 AND X2=X3 AND
X3=X4 THEN IF X1=10 THEN CR=CR+100 :: G
OTO 940 ELSE CR=CR+50 :: GOTO 940
820 IF X1=X2 AND X2=X3 THEN IF X1=10 THEN
 CR=CR+20 :: GOTO 950 ELSE IF X4=10 THE
N CR=CR+25 :: GOTO 950 ELSE CR=CR+10 :: G
OTO 950
830 IF X1=X2 AND X2=X4 THEN IF X1=10 THEN
 CR=CR+20 :: GOTO 950 ELSE IF X3=10 THE
N CR=CR+25 :: GOTO 950 ELSE CR=CR+10 :: G
OTO 950
840 IF X2=X3 AND X3=X4 THEN IF X2=10 THEN
 CR=CR+20 :: GOTO 950 ELSE IF X1=10 THE
N CR=CR+25 :: GOTO 950 ELSE CR=CR+10 :: G
OTO 950
850 IF X1=X3 AND X3=X4 THEN IF X1=10 THEN
 CR=CR+20 :: GOTO 950 ELSE IF X2=10 THE
N CR=CR+25 :: GOTO 950 ELSE CR=CR+10 :: G
OTO 950
860 IF (X1=X2 AND X3=X4)OR(X1=X3 AND X2=X
4)OR(X1=X4 AND X2=X3)THEN 950
870 IF X1=X2 OR X1=X3 OR X1=X4 OR X2=X3 O
R X2=X4 OR X3=X4 THEN CR=CR-5 :: GOTO 9
50
880 IF X1+X2+X3+X4<>10 THEN 930
890 FOR B=5 TO 7 :: IF X1=B OR X2=B OR X3
=B OR X4=B THEN 930
900 NEXT B
910 DISPLAY AT(17,10):"**** COULEURS ****
" :: FOR ZA=110 TO 710 STEP 200 :: FOR
ZB=ZA TO ZA+200 STEP 20 :: CALL SOUND(-1,
ZB,0):: NEXT ZB
920 FOR ZC=ZA+100 TO ZA+300 STEP 20 :: CA
LL SOUND(-1,ZC,0):: NEXT ZC :: NEXT ZA
:: DISPLAY AT(17,10):"" :: CR=CR+100 :: G
OTO 950
930 CR=CR*.75 :: GOTO 950
940 DR=1
950 IF INT(CR)=0 THEN DISPLAY AT(8,12)SIZ
E(13):"" :: GOTO 970
960 DISPLAY AT(8,13)SIZE(5):USING "####":
INT(CR):: IF INT(CR)=1 THEN DISPLAY AT(
8,18)SIZE(7):"FRANC" ELSE DISPLAY AT(8,18
)SIZE(7):"FRANCS"
970 CRC=INT((CR-INT(CR))*100):: IF CRC=0
THEN DISPLAY AT(9,12)SIZE(13):"" :: GOT
O 990
980 DISPLAY AT(9,13)SIZE(3):USING "##":CR
C :: IF CRC=1 THEN DISPLAY AT(9,16)SIZE
(9):"CENTIME" ELSE DISPLAY AT(9,16)SIZE(9
):"CENTIMES"
990 IF CR<=0 OR DR=1 THEN 1030
1000 IF X1=9 OR X2=9 OR X3=9 OR X4=9 THEN
 1010 ELSE 610
1010 DISPLAY AT(13,14):"FLECHES !!" :: DI
SPLAY AT(14,12):"JE RELANCE LE" :: DISP
LAY AT(15,17):"JEU"
1020 FOR B=110 TO 1000 STEP 10 :: CALL SO
UND(-10,B,0):: NEXT B :: DISPLAY AT(13,
14):"" :: DISPLAY AT(14,12):"" :: DISPLAY
 AT(15,17):"" :: X=0 :: GOTO 700
1030 FOR B=0 TO 100 :: NEXT B :: FOR B=5
TO 24 :: DISPLAY AT(B,10)BEEP:"" :: NEX
T B :: CALL SCREEN(8):: DISPLAY AT(5,12):
"LA PARTIE EST"
1040 DISPLAY AT(6,14):"TERMINEE !"
1050 IF CR>MI THEN CRF=CR-MI :: SC(XX,2)=
INT(CR-MI):: DISPLAY AT(8,16):"GAIN:"
1060 IF CR<MI THEN CRF=MI-CR :: SC(XX,3)=
INT(MI-CR):: DISPLAY AT(8,16):"PERTE:"
1070 IF CR=MI THEN DISPLAY AT(8,12):"PART
IE NULLE!" :: GOTO 1150
1080 IF INT(CRF)=0 THEN 1090 ELSE IF INT(
CRF)=1 THEN DISPLAY AT(9,16):"1 FRANC"
ELSE DISPLAY AT(9,13):INT(CRF);"FRANCS"
1090 CRFC=INT((CRF-INT(CRF))*100)
1100 IF CRFC=0 THEN 1110 ELSE IF CRFC=1 T
HEN DISPLAY AT(10,14):"1 CENTIME" ELSE
DISPLAY AT(10,12):CRFC;"CENTIMES"
1110 IF CR>=0 THEN 1150 ELSE DISPLAY AT(1
2,11):"IL ME MANQUE DONC"
1120 IF INT(ABS(CR))=0 THEN 1130 ELSE IF
INT(ABS(CR))=1 THEN DISPLAY AT(13,15):"
1 FRANC" ELSE DISPLAY AT(13,13):INT(ABS(C
R));"FRANCS"
1130 CRFD=INT((ABS(CR)-INT(ABS(CR)))*100)
1140 IF CRFD=0 THEN 1150 ELSE IF CRFD=1 T
HEN DISPLAY AT(14,14):"1 CENTIME !" ELS
E DISPLAY AT(14,12):CRFD;"CENTIMES !"
1150 IF XX=20 THEN 1180 ELSE DISPLAY AT(1
8,11):"UNE AUTRE PARTIE" :: DISPLAY AT(
19,13):"OUI OU NON"
1160 CALL KEY(0,KK,ST):: IF ST=0 OR KK<78
 OR KK>79 THEN 1160
1170 IF KK=79 THEN DR=0 :: GOTO 440 ELSE
1190
1180 FOR B=0 TO 3000 :: NEXT B
1190 CALL DELSPRITE(ALL):: CALL CLEAR ::
CALL CHARSET :: CALL SCREEN(7):: FOR B=
5 TO 7 :: CALL COLOR(B,5,14):: NEXT B ::
DISPLAY AT(1,1):"PARTIE   MISE   GAIN
 PERTE"
1200 IMAGE ##_____####___####___####
1210 FOR B=1 TO XX :: DISPLAY AT(B+1,3):U
SING 1200:B,SC(B,1),SC(B,2),SC(B,3):: N
EXT B :: DISPLAY AT(23,1):"BILAN: MISE
 GAIN    PERTE"
1220 FOR B=1 TO XX :: SC1=SC1+SC(B,1):: S
C2=SC2+SC(B,2):: SC3=SC3+SC(B,3):: NEXT
 B
1230 IF SC2>SC3 THEN GAIN=SC2-SC3 :: PERT
E=0 ELSE IF SC2<SC3 THEN GAIN=0 :: PERT
E=SC3-SC2 ELSE GAIN,PERTE=0
1240 IMAGE ######  ######  ######
1250 DISPLAY AT(24,7):USING 1240:SC1,GAIN
,PERTE
1260 CALL KEY(0,KEY,ST):: IF ST=0 THEN 12
60 ELSE CALL CLEAR :: END
1270 RANDOMIZE :: CA=INT(RND*10)+1 :: RET
URN
1280 SON=(R1-47)*55 :: CALL SOUND(300,SON
,0,SON+50,1,SON+100,2):: RETURN
1290 CALL SOUND(100,400+(CA*50),0):: RETU
RN
1300 FOR B=2 TO 24 :: DISPLAY AT(B,9):""
:: NEXT B :: RETURN
1310 FOR B=7 TO 22 STEP 5 :: CALL HCHAR(B
,9,32):: NEXT B :: RETURN
```

Les nombres en LOGO

Sophie

Nous avons déjà eu amplement l'occasion de manipuler des nombres dans les exemples des articles précédents. Ces nombres servaient à :

- coder (numéro d'appel d'un lutin, numéro de sa forme, code numérique de sa couleur);
- mesurer (des vitesses, par exemple pour les lutins, des longueurs ou des amplitudes de rotations avec la tortue...).

Mais on peut aussi s'en servir tout simplement pour compter.

Un ordinateur, c'est bête, mais pas trop...

Sur le clavier, certaines touches portent les chiffres, et d'autres portent des symboles d'opération ("+", "−", "*" pour la multiplication, et "/" pour la division) et aussi "=", et ".". Attention : prenez l'habitude, pour avoir le plus (+), d'appuyer de la main gauche sur la touche "SHIFT" située à gauche du clavier et de la main droite sur la touche "=". D'abord, c'est une bonne chose que de s'entraîner à utiliser les deux mains, et si possible les dix doigts. Mais il y a plus important : la touche "SHIFT" située à droite du clavier est très proche de la touche "FCTN". Or, FCTN-= vous fait sortir de LOGO, et tout ce que vous aviez emmagasiné dans votre espace de travail est alors perdu, de manière irrémédiable. Quand vous aurez vu partir en un clin d'oeil le travail génial qui vous avait demandé des heures de mise au point, vous comprendrez l'importance de ce conseil d'amie.

Revenons donc à nos moutons, ou plutôt à nos nombres. N'ayez crainte, même si le micro-monde des nombres est un peu plus austère que celui des lutins ou de la tortue, vous ne risquez guère de vous endormir.

C'est au Cours Elémentaire que l'on voit le démarrage de projets sur les nombres. Ayant déjà des habitudes scolaires, les enfants essaient en général d'utiliser la machine comme une calculatrice (ou comme un énoncé d'exercice de calcul), en tapant par exemple :

23+8=

Quelle n'est pas la surprise de voir l'ordinateur faire preuve de mauvaise volonté, et répondre :

PAS ASSEZ DE DONNEES

Certains en déduisent : "c'est comme la chouette (le LITTLE PROFESSOR, ou d'autres calculettes similaires), elle ne fait pas les calculs, mais peut nous dire si on lui donne un résultat juste ou faux". D'autres protestent : "elle n'est pas marrante, ta machine, si c'est nous qui devons faire les calculs !".

Cela engage une série d'activités consistant à préparer des opérations avec leur résultat, et à venir voir l'opinion de la machine. Il se passe encore là quelque chose d'inattendu :

23+8=31

provoque le message :

QUE DOIS-JE FAIRE AVEC VRAI

Remarquons que les élèves ne se découragent pas du tout de recevoir un message :

QUE DOIT-JE FAIRE AVEC FAUX

le cas échéant.

Lorsque les élèves travaillent en groupe, il se met en place très vite une stratégie d'économie que l'on peut exprimer par la règle suivante :

"si j'ai préparé la même opération et trouvé le même résultat que quelqu'un de mon groupe qui est passé à la machine avant moi, je ne réessaie pas" (sous-entendu : on sait bien que la machine ne va pas changer d'avis entre temps).

Par contre, on voit très fréquemment, après qu'un élève ait essayé par exemple :
187+84=231
et obtenu un satisfecit :
QUE DOIS-JE FAIRE AVEC VRAI
plusieurs autres membres du groupe essayer d'autres solutions, telles que :
147+84=221
voire :
147+84=131
et autres, et sembler presque surpris de voir la machine critiquer :
QUE DOIS-JE FAIRE AVEC FAUX

On franchit une étape importante quand l'un des enfants remarque : "Henri a eu VRAI, et moi je n'ai pas le même résultat, alors je n'ai plus de chance, elle va me dire FAUX".
Certains petits malins essaient de se montrer plus astucieux que l'ordinateur, et cherchent le moyen de lui faire faire le calcul. Et contrairement au Basic avec qui "27+18" est effectué, mais reste dans les oubliettes, avec LOGO, on est gratifié d'un rappel à l'ordre ironique :
QUE DOIS-JE FAIRE AVEC 45

Là, c'est la grande stupéfaction : comment, elle est assez maline pour faire le calcul (et même avec les retenues, remarquent certains avec admiration !), et elle demande ce qu'il faut faire avec le résultat ? Vraiment, une grande perte de prestige... pas du tout pour l'ordinateur, qui n'impressionne pas plus la jeune génération qu'un moulin à café électrique, mais plutôt pour les adultes qui font tant d'histoires à propos d'un ordinateur, alors que finalement ce n'est pas grand'chose d'autre qu'un tas de ferraille et de matière plastique, à qui il faut tout dire, même ce qui est évident !
Tout de même, cette réponse incongrue amène certains à réfléchir et à s'interroger : "mais alors, si elle ne sait pas quoi en faire, c'est peut-être qu'elle pourrait en faire autre chose que de l'afficher ?", et à proposer : "elle pourrait peut-être le mettre de côté dans une boîte pour s'en resservir plus tard ?".

Surprise ausi, en primaire, sur les méthodes de calcul de l'ordinateur. En effet, contrairement à l'attente des élèves, "2+3*4" ne donne pas 20 comme résultat (lecture de gauche à droite), mais 14 (utilisation des priorités classiques de l'algèbre). D'où un apprentissage précoce de ces règles par familiarisation grâce à une manipulation volontaire.

Un compte à rebours (où l'on retrouve la récursion)

Nous avions vu, dans l'article précédent, la récursion à propos de la tortue. Nous allons l'utiliser à nouveau pour construire une procédure de compte à rebours.
Que faut-il enseigner à l'ordinateur pour qu'il puisse compter à rebours ? Tout d'abord, il faut afficher la valeur de départ. Autrement dit, notre procédure va ressembler à ceci :

- POUR REBOURS :DEPART
- AFFICHE :DEPART
- FIN

On est évidemment encore loin du compte, car si l'on demande :

- REBOURS 5

on obtient seulement 5.

Autrement dit, il nous faudrait maintenant taper :

- REBOURS 4

c'est-à-dire rappeler la procédure REBOURS avec la valeur :DEPART−1.

Voilà donc l'appel récursif dont nous avons besoin.
Notre procédure devient alors :

- POUR REBOURS :DEPART
- AFFICHE :DEPART
- REBOURS :DEPART−1

Regardons cette nouvelle procédure :
5 4 3 2 1 0 −1 −2 −3 ...

Aïe ! Ca marche un peu trop bien, il nous faut utiliser l'arrêt de panique FCTN-9. En effet, nous n'avons pas indiqué à l'ordinateur à quel moment il doit s'arrêter, et il ne va sûrement pas prendre d'initiative. Or on voudrait que, quand on arrive à zéro, le compte s'arrête. Encore une modification à notre procédure :

- POUR REBOURS :DEPART
- AFFICHE :DEPART
- SI :DEPART=0 ALORS STOP
- REBOURS :DEPART−1
- FIN

Regardons cela :
REBOURS 5
5 4 3 2 1 0
Superbe !

Peut-on faire un compte à rebours de deux en deux ? Peu de changement à apporter à notre précédent raisonnement : il suffit de rappeler la procédure avec :DEPART−2 au lieu de :DEPART−1 ? Faisons une nouvelle procédure :

- POUR REB2 :DEPART
- AFFICHE :DEPART
- SI :DEPART=0 ALORS STOP
- REB2 :DEPART−2
- FIN

Essayons
REB2 10
10 8 6 4 2 0
Trop beau pour être vrai; essayons encore :
REB2 9
9 7 5 3 1 −1 −3 ...

Zut ! Encore un petit pépin, qui vous oblige à avoir recours à nouveau à l'arrêt de panique. Notre test d'arrêt repose sur une égalité, mais celle-ci n'est suceptible de devenir vraie qu'une fois sur deux, en fonction de la valeur de :DEPART. Il nous suffit de la remplacer par une inégalité :

- POUR REB2 :DEPART
- AFFICHE :DEPART
- SI :DEPART<0 ALORS STOP
- REB2 :DEPART−2
- FIN

pour que tout rentre dans l'ordre.

Que se passe-t-il si on modifie l'ordre des lignes d'instructions ?

- POUR REB1 :DEPART
- SI :DEPART=0 ALORS STOP
- AFFICHE :DEPART
- REB1 :DEPART−1
- FIN

(permutation des deux premières lignes de la procédure REBOURS)
REB1 5
5 4 3 2 1

Le zéro n'est pas affiché, puisque le test d'arrêt fait terminer l'exécution avant l'affichage de la valeur zéro.

Et si maintenant on permute les deux dernières lignes ?

- POUR REB :DEPART
- SI :DEPART=0 ALORS STOP
- REB :DEPART−1
- AFFICHE :DEPART
- FIN

on retrouve une procédure avec appel récursif non terminal :
REB 5
1 2 3 4 5
n'a plus rien d'un compte à rebours, puisque l'affichage se fait au moment du dépilage.

Regardons cela de près encore une fois :

REB 5

- La procédure REB s'exécute, :DEPART a la valeur 5. Le test (5=0) a la valeur FAUX, donc cette première ligne est ignorée.
- Il faut alors exécuter REB 4. Cette fois ci, :DEPART vaut 4. Le test (4=0) est encore faux. On trouve donc un nouvel appel.
- REB 3 commence. La valeur de :DEPART est 3. Le test (3=0) est toujours faux. On recommence encore une fois.
- REB 2 s'exécute maintenant, avec la valeur 2 pour :DEPART. Le test (2=0) reste FAUX, donc on a à nouveau un appel.
- REB 1 correspond à la valeur 1 pour :DEPART. Le test (1=0) garde la valeur FAUX, et ceci provoque un nouvel appel à la procédure.
- REB 0 s'exécute alors, avec la valeur 0 pour :DEPART, ce qui donne enfin VRAI pour le test, et amène l'arrêt de cette procédure, qui a, en quelque sorte, terminé son travail et le fait savoir à la procédure qui l'a appelée.
- On se retrouve dans l'exécution de la procédure REB 1, qui s'était arrêtée en cours de route: il reste une ligne à effectuer, l'affichage de la valeur 1, valeur de :DEPART dans cette procédure. La ligne suivante, FIN, indique que cette exécution est terminée.
- Donc REB 2 peut continuer son propre travail, à savoir afficher sa valeur de :DEPART, ici 2. On arrive alors à la FIN de cette procédure.
- REB 3, en affichant sa valeur de :DEPART, soit 3, termine à son tour.
- De même, REB 4 poursuit l'exécution de la ligne d'affichage, et inscrit la valeur 4 qu'a alors :DEPART, dans cette procédure, avant de rendre l'exécution à la procédure appelante.
- REB 5 s'exécute enfin, avec l'affichage de la valeur locale de :DEPART, à savoir 5. Cette procédure terminée, on se trouve ramené au niveau supérieur.

Sortez-les !

Regardons maintenant quelque chose de légèrement différent, avec des procédures pour calculer le CARRE et le CUBE d'un nombre.

Cela semble très simple. Bien sûr, le carré d'un nombre est le produit de ce nombre par lui-même, et le cube d'un nombre est le produit du carré par le nombre (ce qui nous offrira l'occasion d'utiliser une procédure comme sous-procédure d'une autre procédure).

- POUR CARRE :NOMBRE
- AFFICHE :NOMBRE⋆ :NOMBRE
- FIN

semble fonctionner parfaitement :
CARRE 4
16
CARRE 120
14400
CARRE −5
25

Continuons :

- POUR CUBE :VALEUR
- AFFICHE
 CARRE :VALEUR⋆ :VALEUR
- FIN

Là, nous commençons à avoir des ennuis.
Tout d'abord le calcul ne se fait pas comme escompté. Si on demande CUBE 3, le nombre mentionné est 81, qui n'est pas le cube de 3, mais sa puissance quatrième.

Cela provient d'une règle :

Toute opération infixée est effectuée avant les opérations préfixées.

Pour ceux que les qualificatifs infixé et préfixé laissent perplexes, comparons deux façons d'indiquer une addition :

2+3
ou
SOMME 2 3

Dans le premier cas, l'opération est indiquée par un symbole écrit entre les deux nombres sur lesquels porte l'opération. On dit qu'il s'agit d'une notation infixée. Dans le deuxième cas, l'opération est indiquée par un mot qui précède les nombres sur lesquels porte l'opération. On dit qu'il s'agit d'une notation préfixée.

Par conséquent, dans :

- CARRE :VALEUR⋆ :VALEUR

la multiplication (infixée) est effectuée avant l'élévation au carré (préfixée), ce qui explique que l'on trouve la puissance quatrième du nombre fourni.

Deux manières de remédier à cela :

- ou bien parenthéser :
- (CARRE :VALEUR)⋆ :VALEUR
- ou bien inverser l'ordre :
- :VALEUR⋆CARRE :VALEUR

Cette modification ne nous sort pas

complètement de notre problème, car maintenant le système se plaint que le CARRE N'A RIEN SORTI.

Regardons de plus près ce que nous avons mis dans notre procédure CARRE :

- POUR CARRE :NOMBRE
- AFFICHE :NOMBRE⋆ :NOMBRE
- FIN

Non seulement nous avons demandé à la procédure d'effectuer le calcul, mais en plus nous lui avons indiqué qu'elle doit afficher le résultat sur l'écran. Et comme l'ordinateur est bête et discipliné, il nous obéit aveuglément.
Maintenant, essayez de vous imaginer en train de faire de l'affichage (selon vos habitudes, vous vous référerez à du collage d'affiches sur des panneaux électoraux ou à du punaisage de posters sur les murs de votre chambre !). Au début de l'exercice, vous avez en main l'affiche et de quoi la coller (une brosse et un pot de colle, ou des punaises, ou du ruban adhésif et une paire de ciseaux, suivant les circonstances). Quand l'opération est terminée, l'affiche n'est plus entre vos mains, mais sur le mur, et il ne vous reste plus dans les doigts que le nécessaire de fixation.

Hé bien ! C'est exactement ce qui se passe ici. CARRE ayant affiché son résultat, il n'y a plus rien qui puisse être transmis pour le calcul suivant. Et le message, en même temps qu'il vous signale cette anomalie, vous souffle aussi comment vous en sortir ! Il suffira de modifier nos deux procédures :

- POUR CARRE :NOMBRE
- SORS :NOMBRE⋆ : NOMBRE
- FIN

et

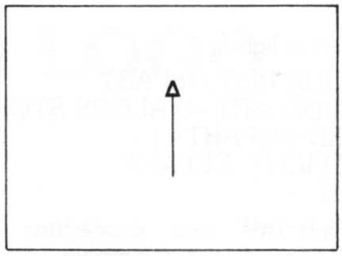
- POUR CUBE :VALEUR
- SORS :VALEUR⋆CARRE : VALEUR
- FIN

Bien entendu, maintenant, exactement comme lorsqu'on demandait directement un calcul, il va falloir penser à préciser ce qu'on souhaite que l'ordinateur fasse avec le résultat, sinon :
CUBE 4
QUE DOIS-JE FAIRE AVEC 64

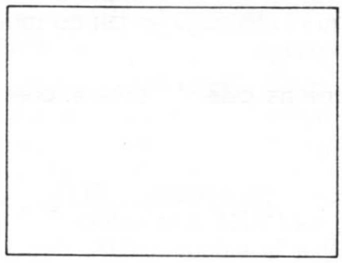
Même si ça vous épate, la tortue sait compter.
On peut, comme nous venons de le faire, jouer avec les nombres pour le plaisir. Mais on peut aussi voir comment la tortue s'en débrouille.

On regarde d'abord l'addition. Deux manières principales de se convaincre que notre petite bête sait ajouter :

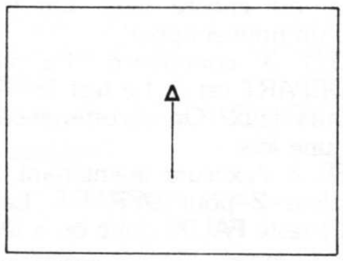
- EFFACEECRAN
- AVANCE 30+25



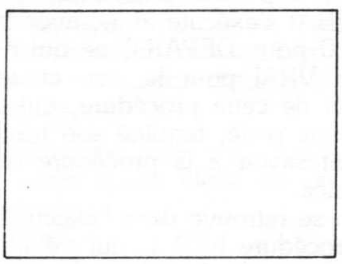
- EFFACEUR
- RECULE 55
- CACHETORTUE



- POSEPLUME
- MONTRETORTUE
- AVANCE 30+25

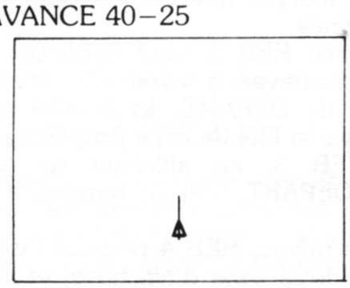


- EFACEUR
- RECULE 30
- RECULE 25
- CACHETORTUE

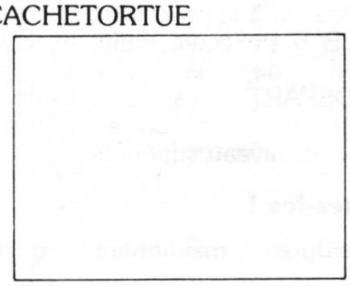


Voyons maintenant avec les soustractions. Au début, cela semble parfait :

- POSEPLUME
- MONTRETORTUE
- AVANCE 40−25



- EFFACEUR
- RECULE 15
- CACHETORTUE



Mais il y a des moments d'inquiétude, quand on essaie :
AVANCE 30−55
car là, oh stupeur ! elle recule.

Première réaction : "la machine est en panne". On recommence, et on s'aperçoit qu'il s'agit là d'un comportement persistant. Même chose pour DROITE 30−120, elle pivote vers la GAUCHE. Il faut donc, pour les élèves de l'école primaire (qui ignorent en général tout des nombres relatifs - les nombres en-dessous de zéro des températures n'ayant pas du tout le même statut que les nombres scolaires -) qui se sont aventurés sur ce terrain glissant, élargir le savoir antérieur pour y incorporer cette acquisition.

Certains en profitent pour poursuivre une exploration sur les nombres :
30−55
ou 20−140
et découvrir ainsi les "nombres à tiret" ou "nombres à moins", et les règles d'opérations qui les concernent.

Bien que la version TI-LOGO ne comporte que les entiers relatifs (de −32767 à 32768), il y a déjà là un vaste micro-monde à explorer. En fait, il est possible d'utiliser dans des calculs des valeurs relationnelles et même des racines carrées, mais cette astuce constitue une autre histoire, que nous vous raconterons une autre fois. Avec les thèmes que nous venons d'aborder ensemble, et avec la fonction aléatoire, on peut déjà s'offrir de quoi meubler agréablement de longues soirées en toute saison.

Sur ce thème des nombres, vous trouverez un chapitre détaillé dans :

Une tortue dans une classe - de Catherine Berdonneau et Rose-Marie Dumas - cahier PACIFIC No 1 - 1982 (le matériel utilisé est un micro-ordinateur qui n'est plus fabriqué depuis plusieurs années, mais les observations restent toujours valables).

Signalons également un compte-rendu d'activités réalisées sur le TI-99/4A avec des enfants en situation d'échec scolaire :

Une semaine d'activités LOGO en C.P.P.N. - de Catherine Berdonneau et Pierre Boyer - Document de travail No 7 - GREPACIFIC - 1983.

Ces deux fascicules sont disponibles auprès de GREPACIFIC - 51 Boulevard des Batignolles - 75008 PARIS (adresse postale uniquement). Le cahier No 1 coûte 60 F, le document de travail No 7 coûte 30 F. Seules les commandes accompagnées d'un chèque à l'ordre de GREPACIFIC en règlement peuvent être traitées.
A bientôt... ■

Graphisme en haute résolution sur le TI-99

Gérard Santraille et Jean-Luc Bazanegue

Notre TI-99 étant dépourvu d'instructions permettant de manipuler simplement, à partir du Basic TI, des graphiques en haute définition (résolution de 256 colonnes et 192 lignes soit 49152 pixels), nous vous présentons un programme en assembleur constitué par un ensemble de routines utilisables séparément à partir du BASIC-TI (avec le module Minimem ou l'Editeur/Assembleur et l'extension de mémoire 32K).

En effet, le simple tracé d'une droite en haute résolution à partir du Basic seul n'est réalisable qu'au moyen de l'instruction CHAR, et seulement après plusieurs pages de calculs. En Basic, pour positionner un point en haute résolution, il faut :

1) déterminer la position du caractère basse résolution (ligne et colonne) où il se positionne;
2) connaître le descripteur du caractère en question (ce qui suppose une gestion complète de tous les descripteurs si l'on travaille en Basic TI, qui ne supporte pas l'instruction "CHARPAT");
3) modifier le descripteur du caractère en "noircissant", dans la matrice 8x8 qui le compose, le pixel (point élémentaire) adéquat.

Cette méthode, on s'en rend aisément compte, est très lourde; elle est de plus extrêmement lente et ne permet d'afficher qu'un nombre limité de points. En effet, le Basic ne permet de redéfinir que 128 caractères; d'autre part, la couleur de chacun de ces caractères n'est pas indépendante.

Plusieurs d'entre vous nous ont proposé un programme Basic permettant, par la méthode que je viens d'exposer, de tracer une courbe en haute résolution. Ces programmes mettent trente minutes pour tracer un simple sinus... et ne peuvent superposer une parabole rouge à un cercle jaune et une ellipse verte... (on pourrait ainsi varier indéfiniment les formes et les couleurs).

Nous vous proposons cependant un petit programme, exécutable en Basic TI avec la console seule, qui peut servir de base à la création d'un programme de tracé plus élaboré. Ce programme permet de sauvegarder sur disquette la courbe obtenue.

```
100 REM ******************
110 REM *   Tracage  de   *
120 REM *   courbes  en   *
130 REM *haute resolution *
140 REM ******************
150 REM *    Basic  TI    *
160 REM ******************
170 REM *    Copyright    *
180 REM *  "99 Magazine"  *
190 REM *Gerard Santraille*
200 REM *   et Jean-luc   *
210 REM *    Bazanegue    *
220 REM ******************
230 REM
240 CALL SCREEN(2)
250 CALL CLEAR
260 PRINT "****************************
   tracage de courbe     **    en haute
resolution   *"
270 PRINT "*definition: 192x256 pixels***
*************************": : : :,"CHO
ISISSEZ"
280 PRINT : :" T)RACER UNE FONCTION": :"
L)IRE UN TRACE": : :
290 CALL SCREEN(15)
300 CALL SOUND(100,700,0)
310 CALL KEY(5,H,S)
320 IF (H-84)*(H-76)THEN 310
330 CALL CLEAR
340 IF H-76 THEN 630
350 PRINT "  Lecture d'un trace.....": :
:
360 INPUT "Nom du fichier : ":F$
370 CALL CLEAR
380 CALL SCREEN(13)
390 GOSUB 1630
400 OPEN #1:F$,INTERNAL,FIXED 19
410 INPUT #1:C$
420 CALL CHAR(33,C$)
430 INPUT #1:C$
440 CALL CHAR(34,C$)
450 INPUT #1:C$
460 CALL CHAR(35,C$)
470 L1=ASC(SEG$(C$,17,1))
480 C1=ASC(SEG$(C$,18,1))
490 CALL HCHAR(L1,1,33,32)
500 CALL VCHAR(1,C1,34,24)
510 CALL HCHAR(L1,C1,35)
520 K=36
530 INPUT #1:C$
540 CALL CHAR(K,C$)
550 L1=ASC(SEG$(C$,17,1))
560 C1=ASC(SEG$(C$,18,1))
570 CALL HCHAR(L1,C1,K)
580 K=K+1
590 IF EOF(1)=0 THEN 530
600 CLOSE #1
610 GOSUB 1670
620 GOTO 620
630 PRINT "  Trace d'une fonction": : :,"
Patientez...."
640 DEF F(X)=EXP(-X/5)*SIN(X)
650 DIM PAT$(127),T(3)
660 FOR I=0 TO 127
670 PAT$(I)="0000000000000000"
680 NEXT I
```

```
690 REF$="0123456789ABCDEF"
700 CALL HCHAR(23,17,32,13)
710 INPUT "Pas du trace : ":P
720 IF (P<.05)+(P>20)THEN 710
730 PRINT : :
740 INPUT "Xmin,Xmax,Xaxe: ":XI,XA,XX
750 IF (XI<XA)*(XX<=XA)*(XX>=XI)THEN 790
760 CALL SOUND(200,200,0)
770 PRINT ,"INCORRECT"
780 GOTO 740
790 INPUT "Fmin,Fmax,Faxe: ":YI,YA,YX
800 IF (YI<YA)*(YX<=YA)*(YX>=YI)THEN 840
810 CALL SOUND(200,200,0)
820 PRINT ,"INCORRECT"
830 GOTO 790
840 CALL CLEAR
850 CALL SCREEN(13)
860 GOSUB 1630
870 DEF HL(X)=INT(1+191*(YA-X)/(YA-YI))
880 DEF HC(X)=INT(1+255*(X-XI)/(XA-XI))
890 DEF Q1(X)=INT((X-1)/8)+1
900 DEF Q2(X)=X-8*(Q1(X)-1)
910 L=HL(YX)
920 C=HC(XX)
930 L1=Q1(L)
940 L2=Q2(L)
950 C1=Q1(C)
960 C2=Q2(C)
970 PAT$(1)=SEG$(PAT$(1),1,2*L2-2)&"FF"&S
EG$(PAT$(1),2*L2+1,14)
980 IF C2>4 THEN 1010
990 R$=STR$(2^(4-C2))&"0"
1000 GOTO 1020
1010 R$="0"&STR$(2^(8-C2))
1020 PAT$(2)=R$&R$&R$&R$&R$&R$&R$&R$
1030 PAT$(3)=SEG$(PAT$(2),1,2*L2-2)&"FF"&
SEG$(PAT$(2),2*L2+1,14)&CHR$(L1)&CHR$(C
1)
1040 CALL CHAR(33,PAT$(1))
1050 CALL CHAR(34,PAT$(2))
1060 CALL CHAR(35,PAT$(3))
1070 CALL HCHAR(L1,1,33,32)
1080 CALL VCHAR(1,C1,34,24)
1090 CALL HCHAR(L1,C1,35)
1100 DD=(XA-XI)*P/256
1110 K=36
1120 U=XI
1130 FOR X=1 TO 256 STEP P
1140 Y=HL(F(U))
1150 U=U+DD
1160 X1=Q1(X)
1170 X2=8*X1-INT(X)
1180 Y1=Q1(Y)
1190 Y2=2*Q2(Y)
1200 IF (Y1<1)+(Y1>24)THEN 1440
1210 CALL GCHAR(Y1,X1,COD)
1220 IF COD>34 THEN 1270
1230 PAT$(K-32)=PAT$(COD-32)
1240 COD=K
1250 IF K=160 THEN 1450
1260 K=K+1
1270 CO$=SEG$(PAT$(COD-32),1,16)
1280 IF X2<4 THEN 1310
1290 X2=X2-4
1300 Y2=Y2-1
1310 S$=SEG$(CO$,Y2,1)
1320 S=POS(REF$,S$,1)-1
1330 T(3)=INT(S/8)
1340 T(2)=INT((S-8*T(3))/4)
1350 T(1)=INT((S-8*T(3)-4*T(2))/2)
1360 T(0)=S-8*T(3)-4*T(2)-2*T(1)
1370 IF T(X2)THEN 1400
1380 S=S+2^X2
1390 S$=SEG$(REF$,S+1,1)
1400 CO$=SEG$(CO$,1,Y2-1)&S$&SEG$(CO$,Y2+
1,16-Y2)
1410 CALL CHAR(COD,CO$)
1420 PAT$(COD-32)=CO$&CHR$(Y1)&CHR$(X1)
1430 CALL HCHAR(Y1,X1,COD)
1440 NEXT X
1450 GOSUB 1670
1460 CALL KEY(3,H,S)
1470 IF H-12 THEN 1460
1480 CALL CLEAR
1490 PRINT "Tapez 'CON' pour poursuivre":
:
1500 BREAK
1510 CALL CLEAR
1520 CALL SCREEN(15)
1530 PRINT "  Sauvegarde du trace...": :
:
1540 INPUT "Nom du fichier : ":F$
1550 OPEN #1:F$,INTERNAL,FIXED 19
1560 FOR I=33 TO K-1
1570 PRINT #1:PAT$(I-32)
1580 NEXT I
1590 CLOSE #1
1600 GOSUB 1670
1610 CALL CLEAR
1620 END
1630 FOR I=1 TO 16
1640 CALL COLOR(I,12,1)
1650 NEXT I
1660 RETURN
1670 FOR I=1 TO 4 STEP .5
1680 CALL SOUND(40,300*I,0)
1690 NEXT I
1700 RETURN
```

Pour s'affranchir de toutes ces limitations, entrons dans le mode "BIT-MAP". Accrochez vos ceintures, le voyage commence...

Sans donner trop de détails sur le fonctionnement propre du programme que nous vous proposons (voir explications pages suivantes et sur le listing du fichier-source), il convient d'éclaircir certains points sur la structure et l'organisation interne de la mémoire du TI-99. Notre ordinateur utilise deux zones de mémoire vive différentes :

1) La mémoire CPU (Central Processing Unit) directement adressable par le microprocesseur. Cette

mémoire se situe physiquement dans la carte d'extension, mis à part une zone de 256 octets située à l'intérieur du microprocesseur et qui constitue, en configuration de base, la seule mémoire directement adressable par le TMS 9900.

2) La mémoire VDP (Video Display Processing), qui est utilisée par le processeur vidéo pour la gestion de l'écran. Il s'agit là de la mémoire résidente de la console.

La mémoire VDP forme un ensemble de 16K contenant notamment la table écran (un tableau de 768 octets indiquant le code ASCII de chaque caractère affiché), la table des couleurs (16 groupes ayant chacun 2 couleurs soit 16 octets), la table de définitions de tous les caractères (512 octets)... etc... le programme Basic. Pour avoir la partition exacte de la mémoire VDP lors d'une utilisation avec l'interpréteur Basic, se référer au manuel de la Mini-mémoire ou à celui de l'Editeur/Assembleur.
La mémoire VDP n'est pas adressée directement par le microprocesseur, ceci explique en partie la relative lenteur du Basic.
Le mode Bit-Map utilise une grande partie de la mémoire vidéo (13056 octets); il n'est donc pas possible de faire cohabiter ce mode d'affichage et le Basic. Donc, lors de la conception du programme, nous étions confrontés au problème suivant : accéder à la haute résolution tout en conservant la souplesse et la facilité d'emploi du Basic.

Nous avons contourné le problème de la façon suivante :

Le graphique est construit point par point dans deux tampons de la mémoire CPU (6K pour la création de l'image et 6K pour la sauvegarde des couleurs). Jusqu'à la visualisation de l'image, le programme Basic reste donc toujours localisé dans la mémoire VDP (qui est, répétons-le, son emplacement normal).
Lors de l'affichage du graphique par une routine (nous verrons plus loin comment l'utiliser), toute la mémoire VDP est sauvegardée dans un autre tampon de 16K en mémoire CPU, et les deux tampons de 6K dont nous avons parlé plus haut sont transférés en mémoire VDP.
L'image apparaît, et le programme Basic n'est pas perdu puisqu'il a été sauvegardé en mémoire CPU.
Lorsque l'image est visualisée, on peut revenir au programme Basic en pressant "BACK" (FCTN-9). Le tampon de 16K réintègre sa position et le programme Basic reprend son cours.
En jonglant ainsi avec les tampons, entre la mémoire CPU et la mémoire VDP, on accède très simplement à la haute résolution.

Voyons maintenant comment utiliser les différentes routines graphiques, et certaines applications des nouvelles capacités de notre TI-99.
Avant toute utilisation de l'une des routines, il convient de charger le programme en Assembleur en mémoire CPU. Si vous possédez le module Editeur/Assembleur, donc un lecteur de disquettes, votre programme doit débuter par :

CALL LOAD("DSK1.GRAPH/OBJ")

à condition, bien sûr, d'avoir choisi "GRAPH/OBJ" comme nom de code objet au moment de l'assemblage.

Note : il n'est pas nécessaire de placer cette instruction dans le programme, on peut très bien l'exécuter en mode commande. Les commandes NEW et BYE ne l'effacent pas puisqu'il est implanté en mémoire CPU.

Si vous n'avez pas de lecteur de disquettes et que vous voulez exploiter cet ensemble de routines à partir de la Mini-mémoire, il convient de charger le programme Basic baptisé "Code", qui "poke" directement le code objet en mémoire. Comme pour le cas précédent, une fois que le programme graphique a été chargé, NEW ou BYE ne l'affecte pas.
La lecture du fichier source (dont le listing est donné plus loin) montre qu'il existe six références externes (étiquettes à partir desquelles on peut lancer l'exécution depuis le Basic). Ces références correspondent à des points d'entrée dans le programme en Assembleur. Dans un but de clarté, nous ne parlerons plus de points d'entrée, mais de routines. Nous allons voir leurs rôles et syntaxes.

EFFT, EFFP, EFFC, AXES, POINT et IMAGE

Routine "EFFT"

Cette routine, qui signifie EFFace Tout, nettoie complètement l'écran graphique. Sa syntaxe est :

CALL LINK("EFFT")

Si vous n'employez pas cette instruction lors de la création d'une nouvelle image, cette dernière se superposera à la précédente. On peut ainsi créer, morceau par morceau, des images graphiques complexes.

Routines "EFFP" et "EFFC"

L'action de "EFFP" (pour EFFace la table des Points) et de "EFFC" (pour EFFace la table des Couleurs) est plus sélective que celle de EFFT :

CALL LINK("EFFP")

nettoie la table de définition de l'image en plaçant la valeur zéro (0) dans tous les octets du tampon de 6K qui définit la position de chaque point.

CALL LINK("EFFC")

a la même action que "EFFP", mais son effet porte sur l'autre tampon de 6K, celui qui définit la couleur des points affichés.

Routine "AXES"

Lors du tracé de fonctions (qu'elles soient cartésiennes, paramétriques ou polaires), il est toujours intéressant de pouvoir placer des axes sur l'écran graphique. Il aurait été possible de dessiner des axes point par point au moyen de la routine "POINT", décrite un peu plus loin, mais nous avons préféré écrire une petite routine en Assembleur qui fait cela automatiquement, permettant ainsi à l'utilisateur de se concentrer uniquement sur la fonction à dessiner.
Nous avons toujours évité de passer des arguments du Basic à l'Assembleur au moyen de LINK (économie de temps et emploi plus souple). Ainsi, quand une routine nécessite des arguments (c'est le cas de "AXES", de "POINT" et de "IMAGE"), il faut les "poker" directement du Basic (par le sous-programme "LOAD") avant d'appeler la routine par "CALL LINK" :

CALL LOAD(−24570,XPOS,YPOS)
CALL LINK("AXES")

Dans le cas présent, la routine "AXES" doit lire deux valeurs, la position de chaque axe. Il convient donc de faire CALL LOAD(−24570,XPOS,YPOS), où XPOS est une valeur comprise entre 0 et 191 (axe des abscisses, horizontal) et YPOS une valeur comprise entre 0 et 255 (axe des ordonnées, vertical).
Ainsi, pour tracer les axes au beau milieu de l'écran, on peut écrire :

CALL LOAD(−24570,95,128)
CALL LINK("AXES")

Petites précisions au sujet des passages de paramètres par "CALL LOAD" :

L'instruction (ou commande) "CALL LOAD" accepte, comme valeur à "poker", tous les nombres compris entre −32768 et 32767, même si ces nombres ne sont pas des entiers. Puisque le paramètre passé doit "tenir" dans un octet (8 bits), la valeur résultant de l'opération sera toujours un entier compris entre 0 à 255.

Exemple :
CALL LOAD(−20000,500.49)
La valeur 500.49 est d'abord réduite à l'entier le plus proche, il nous reste donc 500.

En binaire, et en complément à deux sur un mot mémoire (16 bits soit 2 octets), 500 est représenté de la manière suivante :
500 = 00000001 11110100 (>01F4 en hexa)
A ce stade, l'interpréteur Basic supprime l'octet de poids fort pour ne conserver que l'octet de poids faible. La valeur "pokée" sera donc : 11110100 en binaire, >F4 en hexadécimal, et 244 en décimal.
Si vous désirez obtenir des détails supplémentaires au sujet des mots, octets et des différents codages des nombres, vous pouvez vous reporter à l'article "Les modes d'adressage en Assembleur", paru dans le numéro 3 de "99 Magazine".
Që TI4 2

Pour l'axe horizontal, la routine contrôle la validité de la valeur. Si la position est supérieure à 191, la valeur prise par défaut sera 191 (première ligne de l'écran, en partant du haut).
On peut choisir la couleur des axes tracés en "pokant" une valeur aux adresses −24574 et −24573. En effet, cette adresse (qui, comme nous le verrons plus loin est également utilisée par la routine "POINT") contient la couleur des points "OFF" suivie de celle des points "ON".
Rappelons que le mode Bit-Map ne permet pas de définir de façon totalement indépendante la couleur de chaque pixel. L'écran est découpé en petits tronçons horizontaux de 8 pixels, dont on peut définir la couleur des points "ON" (allumés) et celle des points "OFF" (éteints). La couleur "OFF" se situe à l'adresse −24574 et la couleur "ON" à l'adresse −24573.

ATTENTION : le numéro de la couleur varie entre 0 et 15 et non pas entre 1 et 16 comme en Basic. Ainsi, CALL LOAD(−24574,0,13) indique que le tracé doit se faire en Magenta sur un fond transparent (donc de la couleur de l'écran).

Routine "POINT"

C'est la routine fondamentale de cet ensemble. Elle positionne le point dont les coordonnées sont "pokées" en −24576 pour l'abscisse (comprise entre 0 et 255) et −24575 pour l'ordonnée (comprise entre 0 et 191). Si l'abscisse est supérieure à 191, la routine redonne le contrôle au Basic en ignorant le point.

Comme la routine "AXES", "POINT" va lire aux adresses −24574 et −24573, respectivement, les couleurs "OFF" et "ON" du tronçon de 8 pixels dans lequel se trouve le point en question. Avec cette possibilité, il est possible de superposer plusieurs courbes de couleurs différentes.

La syntaxe est la suivante :

CALL LOAD(−24574,COFF,CON)
pour le choix des couleurs

CALL LOAD(−24576,X,Y)
pour la position du point

CALL LINK("POINT")
pour l'appel de la routine

Note : il est possible de rassembler les deux CALL LOAD :

CALL
LOAD(−24574,X,Y,COFF,CON)

Une fois définies, les couleurs "ON" et "OFF" sont prises par défaut, sauf si l'on "poke" de nouvelles valeurs aux adresses −24574 et −24573. Ainsi :

```
100 CALL LOAD(-24574,0,6)
110 FOR X=0 TO 255
120 CALL LOAD(-24576,X,X/2)
130 NEXT X
```

trace une droite rouge foncé sur fond transparent, donc de la couleur de l'écran. Il est ainsi possible de garder une ou plusieurs valeurs par défaut. Rappelons que ceci n'est possible que parce que les arguments ne sont pas "passés" lors du LINK, mais sont directement "pokés" au moyen de CALL LOAD.

Routine "IMAGE"

Lorsque la construction de l'image est terminée, il convient de l'afficher en la transférant dans la mémoire VDP. Cette routine effectue deux tâches principales :

1) Sauvegarde de toute la mémoire VDP (donc le programme Basic) dans un tampon de 16K de la mémoire CPU.
2) Transfert de l'image (deux tampons de 6K) en mémoire vidéo et visualisation quasi instantanée.

C'est avant d'appeler cette routine que l'on peut choisir la couleur de l'écran qui doit être "pokée" en −24572 (rappelez-vous que la valeur de la couleur doit être comprise entre 0 et 15 et non pas entre 1 et 16).

```
CALL LOAD(-24574,15)
CALL LINK("IMAGE")
```

affiche l'image sur un écran blanc.

Lorsque l'image est visualisée, il est possible de revenir au Basic en pressant "BACK" (FCTN-9).

Voilà... vous savez tout. Votre TI-99 sait désormais tracer en haute résolution.

Quelques remarques et conseils d'utilisation.

L'écran de votre TI-99 n'est pas "orthonormé" : en effet, la hauteur d'un pixel est environ 1,35 fois plus petite que sa largeur. Tenez-en compte lors de vos tracés afin que vos cercles ne ressemblent pas trop à des ellipses.
Pour tracer une série de cercles concentriques (de couleurs différentes) dignes de ce nom, on peut écrire :

```
100 REM ***************
110 REM *   Cercles   *
120 REM *concentriques *
130 REM ***************
140 REM *  Execution:  *
150 REM *    16mn 20s  *
160 REM ***************
170 REM
180 CALL LINK("EFFT")
190 DATA 1,4,6,11,12,13
200 CALL LOAD(-24574,0,1,15)
210 FOR R=10 TO 60 STEP 10
220 READ C
230 CALL LOAD(-24573,C)
240 P=.5/R
250 RY=1.35*R
260 FOR T=0 TO 6.2832 STEP P
270 CALL LOAD(-24576,128+R*COS(T
),95+RY*SIN(T))
280 CALL LINK("POINT")
290 NEXT T
300 NEXT R
310 CALL LINK("IMAGE")
```

N'oubliez pas de charger le programme en assembleur avant de vous servir d'une de ses routines... (ne riez pas, cela nous est souvent arrivé !).
De par la limitation en blocs horizontaux de huit pixels imposée par le découpage, il peut arriver que lorsque deux courbes de couleurs différentes se croisent (ou lorsque une courbe croise un axe), leurs couleurs se confondent (la couleur de la dernière tracée l'emporte).
La relative lenteur de certains tracés (relative si l'on compare au temps nécessaire pour obtenir en Basic des figures à peu près semblables, vous pourrez d'ailleurs faire la comparaison avec le programme Basic que nous vous proposons.) est imputable aux sous-programmes "LINK" et "LOAD". La routine "POINT" est, quant à elle, quasi instantanée.
Pour tracer une fonction exprimée en coordonnées cartésiennes, c'est-à-dire dont l'équation est du type Y=F(X), ou une courbe exprimée en coordonnées paramétriques (X=F(t) et Y=G(t)), il est plus rapide de faire les calculs directement dans le "CALL LOAD" qui "poke" les valeurs de l'abscisse et de l'ordonnée. Le programme suivant trace une rosace selon cette méthode :

```
100 REM ***************
110 REM *    Rosace    *
120 REM ***************
130 REM *  Execution:  *
```

```
140 REM *   13mn 32s   *
150 REM ***************
160 REM
170 CALL LINK("EFFT")
180 CALL LOAD(-24574,0,11,12)
190 FOR T=0 TO 3.1516 STEP .0025
200 CALL LOAD(-24576,128+90*COS(
T)*SIN(5*T),95+60*SIN(T)*SIN(5*T
))
210 CALL LINK("POINT")
220 NEXT T
230 CALL LINK("IMAGE")
```

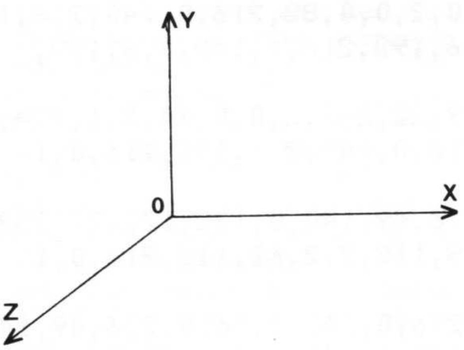
La rosace est exprimée en coordonnées cartésiennes, pour t variant entre 0 et PI, selon :

X=a.Cos(t).Sin(nt)
Y=b.Sin(t).Sin(nt)

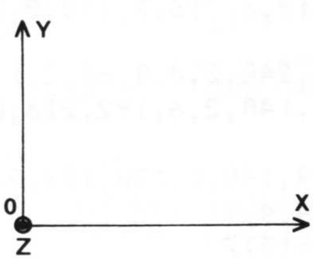
où a et b représentent respectivement la largeur et la hauteur de la rosace et n le nombre de branches.

Note : l'origine de l'écran (0,0) étant située en bas et à gauche (origine classique d'un graphe), lorsque l'on veut centrer une figure géométrique exprimée en coordonnées paramétriques, il convient de rajouter 128 à X et 95 à Y (changement d'origine au centre de l'écran).

Avec cet ensemble de routines, il est possible de simuler la troisième dimension en donnant une impression de profondeur (relief). Pour ce faire, considérons le référentiel direct (qui répond à la règle des trois doigts) suivant :



Cette représentation plane du trièdre "triche": on regarde ce repère sous un certain angle. En effet, dans une représentation rigoureuse, l'axe Z sort de la feuille "en allant vers nous" comme le montre la figure suivante :



Soit une courbe (C), paramétrée selon :

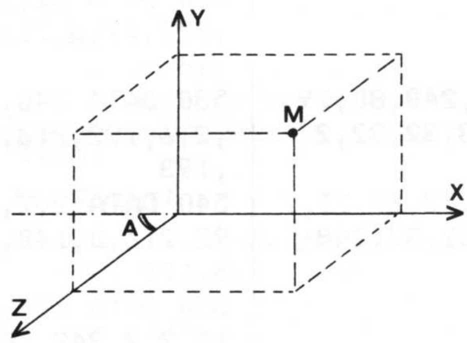
X=f(t)
Y=g(t)
Z=h(t)

L'écran de notre TI ne peut appréhender que X et Y. Pour reporter la cote Z, il convient de passer du référentiel de la figure 2 à celui de la figure 1. Intuitivement, on "voit" bien que plus le dessin sera proche de nous (c'est-à-dire plus Z sera grand), plus il glissera vers le bord inférieur gauche.
Sur le schéma suivant, on voit qu'un point M de coordonnées (x,y,z) sera reporté à la position :

x−z.Cos(A)
y−z.Sin(A)

Nous tenons ici un procédé très simple pour dessiner une courbe en trois dimensions : sa cote intervient dans l'abscisse et l'ordonnée.
Plus l'angle A est important, plus on a l'impression de regarder la courbe "de haut" (ce qui se traduit mathématiquement par un décalage de plus en plus important sur l'axe des ordonnées, puisque le sinus est une fonction croissante jusqu'à PI/2).
Prenons un exemple extrêmement classique : la courbe de LISSAJOU.

Selon le référentiel de la figure 3, l'équation d'une telle courbe est :



x=R.Cos(t)
y=H.Sin(n.t)
z=R.Sin(t)

où R est le rayon de la courbe, H la hauteur et n le nombre de branches.

Supposons maintenant que l'angle sous lequel on regarde le repère soit de 45 degrés (PI/4). Sachant que Cos(45)=Sin(45)=0.707, on traduit la courbe sur une surface plane selon :

x=R⋆(Cos(t)−0.707⋆Sin(t))
y=H⋆Sin(t)−0.707⋆R⋆Cos(t)

Pour que la courbe utilise une grande partie de l'écran, on prendra R=50 (l'unité est le pixel), H=30 et 10 branches. Comme nous l'avons déjà vu, l'écran étant un peu aplati, nous multiplierons y par 1,35. Enfin, pour centrer l'origine du dessin au milieu de l'écran, nous ajouterons 128 à X et 95 à Y. Finalement, l'équation directement exploitable par notre TI-99 est devenue :

x=128+50⋆(Cos(t)−0.707⋆Sin(t))
y=95+1.35⋆30⋆Sin(10t)
−1.35⋆50⋆0.707⋆Cos(t)

soit :

x=128+50⋆Cos(t)−35.35⋆Sin(t)
y=95+40.5⋆Sin(10t) −47.72⋆Cos(t)

Le programme suivant dessine ce motif en vert foncé sur un écran jaune clair (il s'agit de deux couleurs qui s'assemblent d'ailleurs fort bien).

```
100 REM ***************
110 REM *   Lissajou   *
120 REM ***************
130 REM
140 CALL LOAD(-24574,0,12,11)
150 FOR T=0 TO 6.2832 STEP .0025
160 CALL LOAD(-24576,128+50*COS(
T)-35.35*SIN(T),95+40.5*SIN(10*T
)-47.72*COS(T))
170 CALL LINK("POINT")
180 NEXT T
190 CALL SOUND(500,700,0)
200 CALL LINK("IMAGE")
```

Nous vous proposons un dernier programme, qui trace deux ellipses multicolores. Chaque point de ces ellipses est tranformé en une ligne horizontale de trente points. Vous devrez être patient, car il faut un peu plus d'une heure pour obtenir l'image, mais le résultat est intéressant.

```
100 REM ****************
110 REM *   Ellipses   *
120 REM * multicolores *
130 REM ****************
140 REM
150 CALL INIT
160 CALL LOAD(-24572,1)
170 CALL LOAD("DSK1.OBJET")
180 CALL LINK("EFFT")
190 BE1=3.1416/180
200 BE2=BE1*362
210 BE3=BE1/2
220 FOR BE=BE1 TO BE2 STEP BE3
230 X=111+100*COS(BE)
240 Y=90+90*SIN(BE)
250 RANDOMIZE
260 IF X)111 THEN 290
270 CALL LOAD(-24574,0,INT(RND*7
)+2)
280 GOTO 300
290 CALL LOAD(-24574,0,INT(RND*7
)+9)
300 B1=X
310 B2=X+30
320 FOR B=B1 TO B2
330 CALL LOAD(-24576,B,Y)
340 CALL LINK("POINT")
350 NEXT B
360 NEXT BE
```

```
370 FOR BE=BE1 TO BE2 STEP BE3
380 X=111+60*COS(BE)
390 Y=90+60*SIN(BE)
400 RANDOMIZE
410 IF X<=111 THEN 440
420 CALL LOAD(-24574,0,INT(RND*7
)+2)
430 GOTO 450
440 CALL LOAD(-24574,0,INT(RND*7
)+9)
450 B1=X
460 B2=X+30
470 FOR B=B1 TO B2
480 CALL LOAD(-24576,B,Y)
490 CALL LINK("POINT")
500 NEXT B
510 NEXT BE
520 CALL LINK("IMAGE")
```

Vous disposez, avec cet ensemble de routines, d'un outil puissant et surtout très facile à mettre en oeuvre. Nous parlerons, dans de prochains articles, d'autres applications envisageables. En attendant, n'ayez plus aucune retenue graphique et adressez-nous vos plus belles réalisations.

■

```
100 REM ******************
110 REM *   Ce programme  *
120 REM * implante le code *
130 REM * objet en memoire *
140 REM ******************
150 REM * Basic TI + Mini- *
160 REM *     memoire +    *
170 REM *  extension 32K   *
180 REM ******************
190 REM
200 REM pointeur derniere
210 REM adresse libre dans
220 REM la mini-memoire
230 REM
240 CALL LOAD(28702,127,208)
250 REM
260 REM initialisation de
270 REM la table des
280 REM references externes
290 REM
300 DATA 73,77,65,71,69,32,248,248,80,79,
73,78,84,32,248,164,65,88,69,83,32,32,2
48,88
310 DATA 69,70,70,67,32,32,248,72,69,70,7
0,80,32,32,248,54,69,70,70,84,32,32,248
,50
320 FOR B=32720 TO 32767
330 READ V
340 CALL LOAD(B,V)
350 NEXT B
360 REM
370 REM chargement du code
380 REM objet dans
390 REM l'extension 32K
400 REM
410 DATA 225,129,2,128,161,129,6,130,255,
131,3,132,54,133,7,134,17,135,0,0,0,128
,224
420 DATA 129,0,130,12,131,0,132,6,133,0,1
34,7,135,0,0,15,0,7,2,16,1,4,194,2,0,22
4,10
430 DATA 4,240,2,128,248,10,22,252,192,13
0,19,6,2,0,38,118,4,240,2,128,62,118,22
,252
440 DATA 4,96,250,12,200,11,160,8,4,200,2
11,32,160,7,9,140,2,140,0,191,21,4,2,9,
0,191
450 DATA 98,76,16,1,4,201,192,8,192,73,6,
160,248,190,5,136,2,136,1,0,22,248,4,20
1,210
460 DATA 32,160,6,9,136,192,8,192,73,6,16
0,248,190,5,137,2,137,0,192,22,248,194,
224
470 DATA 160,8,4,96,250,12,7,7,208,32,160
,0,9,128,208,160,160,1,9,130,2,130,0,19
1,21
480 DATA 30,2,1,0,191,96,66,4,199,193,1,1
0,84,225,1,2,68,255,7,161,0,2,64,0,7,97
,0,2
490 DATA 3,128,0,11,3,249,3,224,10,209,96
,160,2,4,198,209,160,160,3,11,198,241,7
0,217
500 DATA 5,38,118,193,199,22,1,4,91,4,96,
250,12,216,32,160,4,248,26,2,1,160,10,4
,192
510 DATA 2,2,224,10,216,0,140,2,16,0,216,
0,140,2,16,0,220,96,136,0,128,129,22,25
2,216
520 DATA 32,248,14,131,212,2,1,248,12,192
,177,19,6,216,2,140,2,6,194,216,2,140,2
,16
530 DATA 248,2,0,0,91,2,1,208,0,216,0,140
,2,6,192,216,0,140,2,16,0,216,1,140,0,4
,193
540 DATA 2,2,3,0,2,0,0,88,216,0,140,2,6,1
92,216,0,140,2,6,193,216,1,140,0,6,193,
5,129
550 DATA 128,129,22,249,2,0,0,64,2,1,224,
10,2,2,248,10,216,0,140,2,6,192,216,0,1
40,2
560 DATA 16,0,216,49,140,0,128,129,22,252
,2,0,0,96,2,1,38,118,2,2,62,118,216,0,1
40,2
570 DATA 6,192,216,0,140,2,16,0,216,49,14
0,0,128,129,22,252,216,32,248,10,131,21
2,192
580 DATA 32,248,10,216,0,140,2,6,192,216,
0,140,2,2,224,131,224,6,160,0,14,152,32
,131
590 DATA 117,248,48,22,250,2,224,112,184,
2,1,248,30,192,177,19,6,216,2,140,2,6,1
94,216
600 DATA 2,140,2,16,248,2,0,0,64,2,1,160,
10,2,2,224,10,216,0,140,2,6,192,216,0,1
40,2
610 DATA 16,0,216,49,140,0,128,129,22,252
,4,192,216,0,131,124,4,91
620 FOR B=-2038 TO -1517
630 READ V
640 CALL LOAD(B,V)
650 NEXT B
660 END
```

Sous-programmes en Assembleur

Jean-luc Bazanegue

Nous avons, dans le numéro 3, analysé les différents modes d'adressage disponibles en Assembleur. Cette fois, nous allons travailler sur quelque chose de plus concret : les routines graphiques présentées dans ce numéro.

EFFT, EFFP et EFFC

Ces routines sont constituées par deux segments de programmes. Le premier place la valeur 0 dans les 6 Ko représentant la table des points. Le second réalise la même opération mais, cette fois, dans le tampon de 6 Ko situé dans la partie basse de l'extension 32 Ko. Cette zone de mémoire est destinée à recevoir les informations liées aux couleurs des points.

Puisque nous avons trois points d'entrée et seulement deux segments de programme, il faut utiliser un drapeau (le registre 2) qui nous permettra de savoir si nous avons abordé la routine par "EFFT" ou "EFFP".

Si, depuis le Basic, nous appelons la routine "EFFT", l'instruction SETO place à 1 les seize bits du registre 2. Si nous appelons la routine "EFFP", l'instruction CLR produira l'effet inverse, soit la mise à zéro des seize bits du registre 2. Ainsi, lorsque le programme aura terminé la réinitialisation du tampon "TIMAGE", l'instruction MOV R2,R2 testera l'état de R2 (elle place le bit "égal" du registre d'état à 1 si R2 est égal à zéro). Si R2 est égal à zéro, le programme retourne au Basic, sinon, le tableau "TCOUL" est réinitialisé.

Lors d'un appel de la routine "EFFC", le drapeau n'est pas nécessaire puisque le segment de programme se termine par un branchement vers la routine de retour au Basic.

AXES

La routine "AXES" utilise le registre 11, qui est aussi utilisé pour le retour au Basic. C'est pourquoi nous sommes obligé de sauvegarder le contenu de ce registre dans l'adresse @REG11.

Axe horizontal

Nous commençons par transférer le contenu de l'adresse −24569 dans le registre 12. Comme l'instruction MOVB copie la valeur dans l'octet de poids fort du registre, elle doit être reportée dans l'octet de poids faible. Pour cela, nous avons utilisé l'instruction SRL R12,8 (décalage logique du registre 12, de 8 positions vers la droite) qui place des zéros dans l'octet de poids fort de R12 tout en nous permettant d'obtenir une valeur comprise entre 0 et 255.

Notre écran contient 192 lignes (0 à 191) et un octet peut représenter une valeur comprise entre 0 et 255, il faut donc s'assurer de la validité du paramètre, étant donné qu'une valeur supérieure à 191 provoquerait une écriture à l'extérieur des tampons, donc une destruction partielle des routines, suivie d'un "plantage" à très court terme.

Pour éviter ce genre de surprises, la routine prend 191 comme valeur par défaut si le numéro de ligne est supérieur à 191.

La ligne 0 correspond à la première ligne de l'écran en partant du haut. Pour obtenir un format plus logique (ligne 0 en bas), la valeur représentant la position de l'axe horizontal est inversée.

Une fois cette opération effectuée, les points formant l'axe sont placés dans le tampon "TIMAGE" par la routine "POINT".

Axe Vertical

Mis à part le contrôle de validité et l'inversion de valeur, inutiles pour l'axe vertical, ce segment de programme fonctionne de la même façon que pour l'axe horizontal.

Avant de revenir au Basic, il faut placer l'adresse de retour dans le registre 11. Puisque nous l'avions sauvegardée dans l'adresse @REG11, l'instruction MOV @REG11,R11 réalise cette tâche très simplement.

POINT

La routine "POINT" peut être appelée à partir du Basic ou de la routine "AXES", nous avons donc utilisé, comme pour les routines "EFFT" et "EFFP", un drapeau permettant de savoir par quel point la routine a été abordée.

Si, lorsque la routine est appelée du Basic, la valeur de Y est supérieure à 191 (CI R2,191), le point à afficher est ignoré et la routine effectue un retour immédiat au programme appelant. Si la valeur est inférieure ou égale à 191, elle est inversée, tout comme dans la routine "AXES".

Le segment de programme qui calcule l'octet et le bit à modifier dans le tampon "TIMAGE" est celui que l'on retrouve dans tous les logiciels utilisant le mode Bit-map, il est donc inutile d'insister sur ce point.

IMAGE

Dans cette routine, les seuls points délicats sont les lectures et écritures en mémoire vive vidéo, sans utiliser VSBR, VMBR, VSBW, VMBW et VWTR. Nous avons préféré cette solution un peu plus complexe à mettre en oeuvre, mais qui permet de diminuer le temps d'exécution de la routine d'environ 50%.

Ecriture d'un octet en mémoire vive vidéo

Pour écrire un octet, il faut :

- placer à 1 le bit 1 de l'adresse dans laquelle vous désirez écrire. Par exemple, si vous voulez écrire un octet à l'adresse >2000 (00100000 00000000), cette adresse devient >6000 (01100000 00000000);
- transférer l'octet de poids faible de l'adresse dans laquelle on veut écrire dans l'adresse >8C02;
- provoquer un délai avec, par exemple, SWPB ou NOP (JMP $+1);
- transférer l'octet de poids fort de l'adresse dans laquelle on veut écrire dans l'adresse >8C02;
- provoquer à nouveau un délai;
- transférer la valeur à écrire dans l'adresse >8C00.

Lecture d'un octet en mémoire vive vidéo

La méthode est la même que pour l'écriture, sauf pour les points suivants :

- le bit 1 de l'adresse ne doit pas être placé à 1;
- au lieu de transférer une valeur dans l'adresse >8C00, il faut lire une valeur à l'adresse >8800

Modification d'un registre vidéo

Pour modifier un registre, il faut :

- transférer l'octet contenant le numéro du registre dans l'adresse >8C02. Le bit 0 de l'octet doit être placé à 1. Par exemple, pour modifier le registre >7 (00000111), la valeur >87 (10000111) doit être transférée;
- provoquer un délai;
- transférer l'octet contenant la valeur à charger dans le registre vers l'adresse >8C02.

Lecture ou écriture de plusieurs octets en mémoire vive vidéo

L'adresse >8C02 étant auto-incrémentée, il n'est pas nécessaire de donner à chaque fois l'adresse dans laquelle on désire lire ou écrire. Par exemple, si une valeur a été écrite dans l'adresse >0001, l'adresse >0002 est automatiquement pointée.

Il est ainsi possible de modifier ou de lire jusqu'à un maximum de 16 Ko en indiquant une seule adresse. ■

```
****************************************
*                                      *
* ROUTINES GRAPHIQUES - HAUTE RESOLUTION *
*                                      *
****************************************
* Copyright "99 Magazine", Gerard Santraille *
* et Jean-Luc Bazanegue                *
****************************************

       DEF   EFFT,EFFP,EFFC,AXES,POINT,IMAGE  * References externes.

       AORG >A000  * Origine absolue.

COORD  DATA 0      * Coordonnees  -  X dans -24576  -  Y dans -24575.
COULP  DATA 0      * Couleurs - points "off" dans -24574
*                  *          - points "on" dans -24573.
COULE  DATA 0      * Couleur de l'ecran dans -24572.
POSAXE DATA 0      * Positions des axes -  Y dans -24570  -  X dans -24569.
REG11  DATA 0      * 2 octets pour la sauvegarde du registre 11 (retour Basic).
TBASIC BSS  >4000  * Tampon pour sauvegarde de la memoire vive video (16K).
TIMAGE BSS  >1800  * Tampon pour la creation de l'image en memoire vive (4K).
RVDP1  DATA >E181  * Registre VDP 1 en mode BIT-MAP (affichage inhibe).

REGVDP DATA >0280,>A181,>0682,>FF83    * Registres VDP pour le mode BIT-MAP.
       DATA >0384,>3685,>0786,>1187,0  *
RVDPBA DATA >0080,>E081,>0082,>0C83    * Registres VDP pour le Basic
       DATA >0084,>0685,>0086,>0787,0  * (valeurs par defaut).

TCOUL  EQU  >2676  * Tampon pour la determination des couleurs (4K).
ETAT   EQU  >837C  * Adresse du registre d'etat de l'interpreteur GPL.
RCLAV  EQU  >8375  * Adresse contenant le code de la touche pressee.
BACK   BYTE 15     * Code de la touche "BACK" (FCTN-9).
       EVEN        * Pour continuer l'assemblage sur une adresse paire.

**************************************************************
*                                                            *
* Routine "EFFT" : efface les tampons "TIMAGE" et "TCOUL".   *
*    "    "EFFP" : efface le tampon "TIMAGE".                *
*    "    "EFFC" : efface le tampon "TCOUL".                 *
* Appel du Basic par : - CALL LINK("EFFT").                  *
*                      - CALL LINK("EFFP").                  *
*                      - CALL LINK("EFFC").                  *
*                                                            *
**************************************************************

EFFT   SETO R2               * R2 = drapeau haut si appel par CALL LINK("EFFT").
       JMP  EFFP2            * Saut vers effacement des points.
EFFP   CLR  R2               * R2 = drapeau bas si appel par CALL LINK("EFFP").
EFFP2  LI   R0,TIMAGE        * Charge l'adresse du tampon "TIMAGE" dans R0.
EFFP1  CLR  *R0+             * Met a zero le mot dont l'adresse est contenue
*                            * par le registre R0. Incremente R0 par 2.
       CI   R0,TIMAGE+>1800  * Test si fin de tampon.
       JNE  EFFP1            * Si non, poursuivre la mise a zero.
       MOV  R2,R2            * Controle de l'etat du drapeau.
       JEQ  FEFF             * Si bas, retour au Basic.
EFFC   LI   R0,TCOUL         * Charge l'adresse du tampon "TCOUL" dans R0.
EFFC1  CLR  *R0+             * Met a zero le mot dont l'adresse est contenue
*                            * par le registre R0. Incremente R0 par 2.
       CI   R0,TCOUL+>1800   * Test si fin de tampon.
       JNE  EFFC1            * Si non, poursuivre la mise a zero.
FEFF   B    @RETOUR          * Vers routine de retour au Basic.

************************************************************************
*                                                                      *
* La routine "AXES" realise le trace des axes en fonction des adresses *
* -24570 : axe vertical (Y). -24569 : axe horizontal (X).              *
* Couleurs des axes: - points "off" dans -24574                        *
*                    - points "on" dans -24573.                        *
*                                                                      *
************************************************************************

AXES   MOV  R11,@REG11       * Sauvegarde de l'adresse de retour au Basic.

*******************
* Axe horizontal *
*******************
```

```
        CLR  R8              * Zero dans R8.
        MOVB @POSAXE+1,R12   * Position dans l'octet de poids fort de R12.
        SRL  R12,8           * Decalage logique vers la droite de 8 positions.
        CI   R12,191         * Compare le contenu de R12 avec 191.
        JGT  AXEY1           * Si superieur, saut a AXEY1.
        LI   R9,191          * Sinon, charge la valeur 191 dans R9.
        S    R12,R9          * Soustrait R12 a R9. La position reelle est dans R9.
        JMP  AXEY            * Vers trace ligne horizontale.
AXEY1   CLR  R9              * Ligne horizontale = 1ere ligne en bas de l'ecran.
AXEY    MOV  R8,R0           * Copie de R8 dans R0.
        MOV  R9,R1           * Copie de R9 dans R1.
        BL   @POINT2         * Appel de la routine pour calculer un point.
        INC  R8              * Point suivant.
        CI   R8,256          * Verification si tous les points sont calcules.
        JNE  AXEY            * Si non, continuer.

****************
* Axe vertical *
****************

        CLR  R9              * Zero dans R9.
        MOVB @POSAXE,R8      * Position dans l'octet de poids fort de R8.
        SRL  R8,8            * Decalage logique vers la droite de 8 positions.
AXEX    MOV  R8,R0           * Copie de R8 dans R0.
        MOV  R9,R1           * Copie de R9 dans R1.
        BL   @POINT2         * Appel de la routine pour calculer un point.
        INC  R9              * Point suivant.
        CI   R9,192          * Verification si tous les points sont calcules.
        JNE  AXEX            * Si non, continuer.

        MOV  @REG11,R11      * Place l'adresse de retour au Basic dans R11.
        B    @RETOUR         * Branchement sur la routine de retour au Basic.

*************************************************************************
*                                                                       *
* La routine "POINT" calcule l'octet et le bit a modifier dans le tampon *
* "TIMAGE" en fonction des coordonnees X et Y transmises par le         *
* programme Basic aux adresses >A000 et >A001.                          *
* Les couleurs des points "off" et "on" doivent etre aux adresses       *
* -24574 et -24573.                                                     *
* appel a partir du Basic par CALL LINK("POINT").                       *
*                                                                       *
*************************************************************************

POINT   SETO R7             * R7 = drapeau haut si appel a partir du Basic.
        MOVB @COORD,R0      * Copie le contenu de l'adresse >A000 dans l'octet de
*                           * poids fort de R0.
        SRL  R0,8           * Decale le contenu de R0 de 8 positions vers la droite
        MOVB @COORD+1,R2    * Copie le contenu de l'adresse >A001 dans l'octet de
*                           * poids fort de R2.
        SRL  R2,8           * Decale le contenu de R2 de 8 positions vers la droite
        CI   R2,191         * Compare le contenu de R2 (Y) avec 191.
        JGT  FHPLOT         * Si Y est superieur a 191, retour au Basic.
        LI   R1,191         * Charge la valeur 191 dans le registre 1.
        S    R2,R1          * Soustrait la valeur contenu dans le registre 2 a
*                           * R1 pour inverser Y. Y se trouve maintenant dans R1.
POINT2  CLR  R7             * R7 = drapeau bas si appel a partir de "AXES".
        MOV  R1,R4          * Copie le contenu de R1 dans R4.
        SLA  R4,5           * Decale le registre 4 de 5 positions vers la gauche.
        SOC  R1,R4          * Force a 1 les bits de R4 qui correspondent aux bits
*                           * a 1 de R4.
        ANDI R4,>FF07       * Intersection logique entre le registre 4 et >FF07.
        A    R0,R4          * Ajoute R0 a R4.
        ANDI R0,7           * Intersection logique entre le registre 0 et 7.
        S    R0,R4          * Soustrait R0 a R4.
*                           * R4 contient l'adresse de l'octet a modifier, par
*                           * rapport a l'adresse de base du tampon "TIMAGE".
*                           * R0 contient la position du bit a modifier dans
*                           * l'octet obtenu par R4.
        LI   R3,>8000       * Charge la valeur >8000 (1000000000000000 en binaire
*                           * dans R3.
        SRC  R3,0           * Decalage circulaire vers la droite de R3 du nombre
*                           * de positions contenu dans les 4 bits de droite de R0.
        SOCB R3,@TIMAGE(R4)     * Force a 1 le bit correspondant au bit a 1
*                               * du registre 3.
        MOVB @COULP,R5          * Couleur des points "off" dans R5.
        CLR  R6                 * Registre 6 = 0.
        MOVB @COULP+1,R6        * Couleur des points "on" dans R6.
        SRC  R6,12              * Decalage circulaire de 12 positions a droite.
        SOCB R6,R5              * Place a 1 les bits de R5 correspondants aux
*                               * bits a 1 de R6.
        MOVB R5,@TCOUL(R4)      * Transfere le contenu de l'octet de poids fort de
*                               * R5 dans l'octet situe a l'adresse @TCOUL +
*                               * la valeur contenue par R4.
```

```
       MOV  R7,R7               * Test de l'etat du drapeau.
       JNE  FHPLOT              * Si haut, retour au Basic.
       RT                       * Sinon, retour a la routine "AXES".
FHPLOT B    @RETOUR             * Branchement sur la routine de retour au Basic

**************************************************************************
*                                                                        *
* La routine "IMAGE" sauvegarde le contenu de la memoire vive video dans le *
* tampon de 16K baptise  "TBASIC", initialise les registres VDP pour le   *
* mode BIT-MAP et affiche l'image en transferant le contenu du tampon    *
* "TIMAGE" dans la memoire vive video, a partir de l'adresse >0000, et le *
* tampon "TCOUL" a partir de l'adresse >2000.                            *
* Appel a partir du Basic par CALL LINK("IMAGE")                         *
*                                                                        *
**************************************************************************

IMAGE  MOVB @COULE,@REGVDP+14  * Copie de l'adresse -24572 (couleur de l'ecran).

****************************************
* Sauvegarde de la memoire vive video *
****************************************

       LI   R1,TBASIC           * Adresse du tampon en memoire vive du CPU.
       CLR  R0                  * Adresse du tampon en memoire vive video.
       LI   R2,TBASIC+>4000     * Nombre d'octet a transferer.
       MOVB R0,@>8C02      *
       NOP                  *
       MOVB R0,@>8C02       *
       NOP                      * Remplace la routine VMBR.
SB     MOVB @>8800,*R1+     *
       C    R1,R2           *
       JNE  SB             *

**********************************************************
* Initialisation des registres video pour le mode BIT-MAP *
**********************************************************

       MOVB @REGVDP+2,@>83D4    * Copie du registre VDP 1 dans >83D4.
*                               * (affichage inhibe).
       LI   R1,REGVDP           * Adresse des valeurs pour les registres video.
IRBM   MOV  *R1+,R2        *
       JEQ  ITLUT           *
       MOVB R2,@>8C02          * Remplace la routine VWTR.
       SWPB R2                  *
       MOVB R2,@>8C02       *
       JMP  IRBM           *

********************************************************************
* Initialisation table des lutins (pour eviter les images "fantomes")*
********************************************************************

ITLUT  LI   R0,>005B        * Adresse en memoire vive video.
       LI   R1,>D000        * D0 = valeur a ecrire.
       MOVB R0,@>8C02 *
       SWPB R0         *
       MOVB R0,@>8C02  * Remplace VSBW.
       NOP             *
       MOVB R1,@>8C00 *

*****************************************************************
* Initialisation de la table image ecran (>00 a >FF trois fois) *
*****************************************************************

       CLR  R1             *
       LI   R2,768          *
       LI   R0,>0058        *
       MOVB R0,@>8C02       *
       SWPB R0              *
       MOVB R0,@>8C02        * Remplace VSBW.
ITE    SWPB R1              * Voir article pour explications.
       MOVB R1,@>8C00       *
       SWPB R1              *
       INC  R1              *
       C    R1,R2           *
       JNE  ITE            *

*******************************************************
* Transfert du tampon "TIMAGE" en memoire vive video *
*******************************************************

       LI   R0,>0040            * Adresse de la table des definitions.
       LI   R1,TIMAGE           * Adresse du tampon en memoire vive du CPU.
       LI   R2.TIMAGE+>1800     * Derniere adresse du tampon.
```

```
        MOVB R0,@>8C02        *
        SWPB R0                *
        MOVB R0,@>8C02         *
        NOP                     * Remplace VMBR. (voir article).
TTI     MOVB *R1+,@>8C00       *
        C    R1,R2              *
        JNE  TTI               *

*************************************************************
* Transfert du tampon "TCOUL" en memoire vive video *
*************************************************************

        LI   R0,>0060          *
        LI   R1,TCOUL           *
        LI   R2,TCOUL+>1800     *
        MOVB R0,@>8C02          *
        SWPB R0                 * Meme methode que pour
        MOVB R0,@>8C02          * le tampon "IMAGE".
        NOP                     *
TTC     MOVB *R1+,@>8C00        *
        C    R1,R2              *
        JNE  TTC               *

*********************************
* Retablissement de l'affichage *
*********************************

        MOVB @RVDP1,@>83D4
        MOV  @RVDP1,R0
        MOVB R0,@>8C02
        SWPB R0
        MOVB R0,@>8C02

**************************
* Scrutation du clavier *
**************************

        LWPI >83E0             * Memoire de travail de l'interpreteur GPL.
CLAV    BL   @>000E
        CB   @RCLAV,@BACK      * Controle si la touche "BACK" est pressee.
        JNE  CLAV              * Si non, continue la scrutation.
        LWPI >20BA             * Charge la memoire de travail utilisateur (USRWSP).
*                              * NOTE: Pour une utilisation de cette routine avec le
*                              * module Mini-Memoire, il faut remplacer la
*                              * valeur >20BA par >70B8.

*************************************************************************
* Reinitialisation des registres video pour le Basic (valeurs par defaut)*
*************************************************************************


        LI   R1,RVDPBA       *
IRB     MOV  *R1+,R2          *
        JEQ  TTBAS            *
        MOVB R2,@>8C02        * Meme methode que pour le mode BIT-MAP.
        SWPB R2               *
        MOVB R2,@>8C02        *
        JMP  IRB             *

*****************************************************************
* Memoire vive video comme avant l'appel de la routine "IMAGE" *
*****************************************************************

TTBAS   LI   R0,>0040           * Adresse du tampon en memoire vive video.
        LI   R1,TBASIC          * Adresse du tampon en memoire vive du CPU.
        LI   R2,TBASIC+>4000    * Derniere adresse du tampon.
        MOVB R0,@>8C02         *
        SWPB R0                 *
        MOVB R0,@>8C02          *
        NOP                     * Remplace  VMBW.
TTB     MOVB *R1+,@>8C00        *
        C    R1,R2              *
        JNE  TTB               *

*****************************
* Retour au programme Basic *
*****************************

RETOUR  CLR  R0              * Zero dans le registre 0.
        MOVB R0,@ETAT        * Zero dans le registre d'etat de l'interpreteur GPL.
        RT                   * Equivalent a : B *R11.
        END
```

Horloge

Christian Monfort

Ce programme, écrit en **Basic Etendu**, transforme l'écran de votre TI en horloge - chronomètre - calendrier.

Le démarrage de l'horloge se fait après avoir entré la date (en chiffres).

Les aiguilles sont remplacées par :

- la coloration en bleu du chiffre indiquant l'heure en cours;
- une grosse étoile rouge indiquant les minutes;
- une petite étoile blanche qui tourne au rythme des seconde.

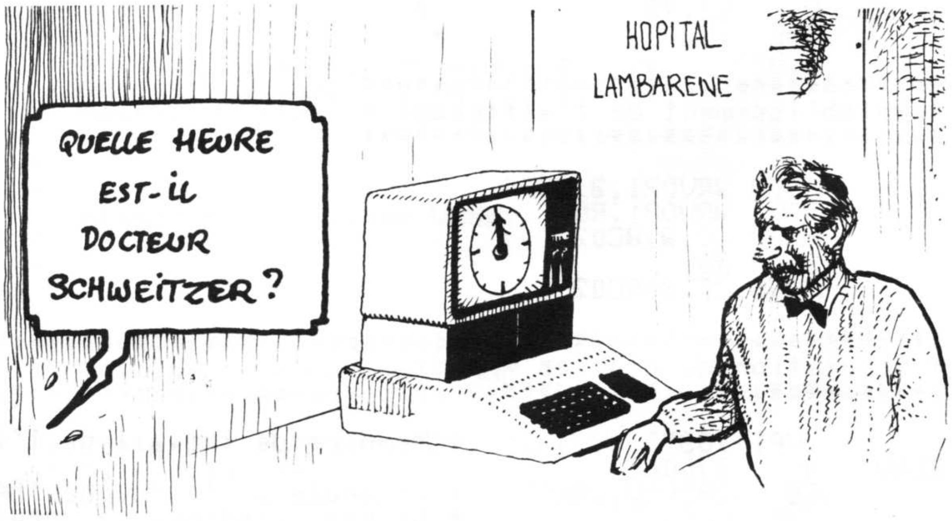
Le chronomètre démarre, s'arrête et s'efface en appuyant sur une touche quelconque. Ces opérations n'étant prises en compte qu'au passage sur une instruction CALL KEY, il faut maintenir le doigt sur la touche jusqu'à ce que l'on entende le signal sonore.

Le calendrier change de date lorsque l'horloge marque 24 heures, ce qui est la moindre des choses. Les années bissextiles sont prévues.

Cette horloge n'est pas d'une grande utilité. Elle montre (sans jeu de mots) cependant la façon de faire tourner non pas des aiguilles, mais des points les matérialisant en fonction du temps.

Note : si vous utilisez une extension 32K, il est possible que l'horloge prenne rapidement de l'avance. Dans ce cas, vous devrez modifier les lignes 1000 et 1010 en conséquence (variable DELAI).

NDLR *: comme prévu, Christian Monfort se voit attribuer un abonnement pour quatre numéros, avec cassettes.* ■




```
100 REM ******************
110 REM *     Horloge      *
120 REM ******************
130 REM *  Basic  Etendu  *
140 REM ******************
150 REM *    Copyright     *
160 REM *Christian Monfort*
170 REM * et "99 Magazine"*
180 REM ******************
190 CALL CLEAR :: CALL SCREEN(4):: ON WAR
NING NEXT
200 DISPLAY AT(3,7):"**************":"
  *              *":"      *   HORLOGE
*"
210 DISPLAY AT(6,7):"*              *":"
   * CALENDRIER *":"              *
*":"       **************"
220 P=1
230 R=8 ! RETARD AU DEPART
240 DISPLAY AT(18,1):" Pour demarrer,arre
ter,ou":"mettre a zero le chronometre":
"il suffit d'enfoncer une     ":"touche et
 de la relacher des"
250 DISPLAY AT(22,1):"que l'on entend le
signal."
260 FOR Q=1 TO 1000 :: NEXT Q
270 FOR Q=18 TO 22 :: DISPLAY AT(Q,1):: N
EXT Q
280 DISPLAY AT(14,4)BEEP:"JOUR (JJ) ?  :"
 :: ACCEPT AT(14,19)SIZE(2)VALIDATE(DIG
IT):JR
290 IF JR>31 THEN 280
300 DISPLAY AT(16,4)BEEP:"MOIS (MM) ?  :"
 :: ACCEPT AT(16,19)SIZE(2)VALIDATE(DIG
IT):MO :: IF MO>12 OR MO<1 THEN 300
310 DISPLAY AT(18,4)BEEP:"ANNEE(AAAA)? :"
 :: ACCEPT AT(18,19)SIZE(4)VALIDATE(DIG
IT):AN
320 DISPLAY AT(20,4)BEEP:"HEURE DE DEPART
  ? :" :: ACCEPT AT(20,25)SIZE(2)VALIDA
TE(DIGIT):H
330 IF H>24 THEN 320
340 IF H>12 THEN H=H-12 :: X=1
350 IF H=0 THEN H=12
360 A=H
370 DISPLAY AT(22,4)BEEP:"MINUTE DE DEPAR
T ? :" :: ACCEPT AT(22,25)SIZE(2)VALIDA
TE(DIGIT):M
380 IF M>59 THEN 370
390 DISPLAY AT(24,4)BEEP:"SECONDE DE DEPA
RT? :" :: ACCEPT AT(24,25)SIZE(2)VALIDA
TE(DIGIT):S
400 IF S>59 THEN 390
410 M1=PI/2-(M*PI*6/180)
420 S=S+R :: S1=PI/2-(S*PI*6/180)
430 AV=INT(AN/100):: AS=AN-100*AV
440 IF AS<>0 AND(AS-4*INT(AS/4))=0 OR AS=
0 AND(AV-4*INT(AV/4))=0 THEN FEV=30 ELS
E FEV=29
450 GOSUB 1270
460 CALL CLEAR
470 CALL CHAR(33,"0000241830C1824"):: CALL
 CHAR(34,"00422418FF182442")
480 REM DESSIN DES CHIFFRES
490 CALL CHAR(38,"0000001818l8")
500 CALL CHAR(40,"1010101010101010")
510 CALL CHAR(35,"FF0101FF808080FF")
520 CALL CHAR(47,"FF0101FF010101FF")
530 CALL CHAR(64,"818181FF01010101")
540 CALL CHAR(72,"FF8080FF010101FF")
550 CALL CHAR(80,"FF8080FF818181FF")
560 CALL CHAR(88,"FF01010101010101")
570 CALL CHAR(96,"FF8181FF818181FF")
580 CALL CHAR(107,"FF8181FF010101FF")
590 CALL CHAR(119,"1010101010101010")
600 CALL CHAR(113,"FF818181818181FF")
610 CALL CHAR(120,"1010101010101010")
620 CALL CHAR(128,"1010101010101010")
630 CALL CHAR(129,"FF0101FF808080FF")
640 REM MISE EN PLACE               DES C
HIFFRES
650 T=3*PI/2+PI/6 :: CALL SPRITE(#1,40,2,
88+88*SIN(T),120+88*COS(T))
660 T=T+PI/6 :: CALL SPRITE(#2,35,2,88+88
*SIN(T),120+88*COS(T))
670 T=T+PI/6 :: CALL SPRITE(#3,47,2,88+88
*SIN(T),120+88*COS(T))
680 T=T+PI/6 :: CALL SPRITE(#4,64,2,88+88
*SIN(T),120+88*COS(T))
690 T=T+PI/6 :: CALL SPRITE(#5,72,2,88+88
*SIN(T),120+88*COS(T))
700 T=T+PI/6 :: CALL SPRITE(#6,80,2,88+88
*SIN(T),120+88*COS(T))
710 T=T+PI/6 :: CALL SPRITE(#7,88,2,88+88
*SIN(T),120+88*COS(T))
720 T=T+PI/6 :: CALL SPRITE(#8,96,2,88+88
*SIN(T),120+88*COS(T))
730 T=T+PI/6 :: CALL SPRITE(#9,107,2,88+8
8*SIN(T),120+88*COS(T))
740 T=T+PI/6 :: CALL SPRITE(#13,113,2,88+
88*SIN(T),120+88*COS(T))
750 CALL SPRITE(#10,119,2,88+88*SIN(T),12
0+88*COS(T)-8,0,0)
760 T=T+PI/6 :: CALL SPRITE(#11,120,2,88+
88*SIN(T),120+88*COS(T),0,0)
770 CALL SPRITE(#14,120,2,88+88*SIN(T),12
0+88*COS(T)-8)
780 T=T+PI/6 :: CALL SPRITE(#15,129,2,92+
88*SIN(T),120+88*COS(T))
790 CALL SPRITE(#12,128,2,92+88*SIN(T),12
0+88*COS(T)-8)
800 CALL COLOR(#A,5):: IF H>9 THEN CALL C
OLOR(#A+3,5)
810 CALL SPRITE(#16,33,16,88-72*SIN(S1),1
20+72*COS(S1))
820 CALL SPRITE(#17,34,14,88-64*SIN(M1),1
20+64*COS(M1))
830 W=18
840 FOR T1=11*PI/6 TO 0 STEP -PI/6 :: CAL
L SPRITE(#W,38,13,88-72*SIN(T1),120+72*
COS(T1)):: W=W+1 :: IF W>28 THEN 860
850 NEXT T1
860 CALL HCHAR(12,25,38):: CALL COLOR(1,1
3,1)
870 IF JR<10 THEN JR$="0"&STR$(JR)ELSE JR
$=STR$(JR)
880 DISPLAY AT(14,11):JOUR$ :: DISPLAY AT
(15,14):JR$ :: DISPLAY AT(16,10):MOI$
890 FOR S2=S TO 59
900 REM CHRONOMETRE
910 CALL KEY(0,K,STATE)
920 IF STATE=0 THEN 940 ELSE CALL SOUND(1
00,110,0)
930 IF C+D<>0 THEN 960 ELSE 980
940 IF C+D=0 THEN 990
950 IF AR=1 THEN 1000 ELSE 980
960 IF AR<>1 THEN 970 ELSE AR=0 :: C=0 ::
 D=0 :: GOTO 1000
970 AR=1 :: GOTO 1000
980 C=C+1 :: IF C=60 THEN C=0 :: D=D+1
990 AR=0
1000 IF C+D=0 THEN DELAI=121 ELSE DELAI=9
2
1010 IF AR=1 THEN DELAI=98
1020 FOR DEL=1 TO DELAI :: NEXT DEL
1030 IF C+D<>0 THEN DISPLAY AT(10,14-LEN(
STR$(D)))SIZE(5):STR$(D);".";STR$(C)ELS
E DISPLAY AT(10,12)
1040 S1=S1-(PI/30)
1050 CALL LOCATE(#16,88-72*SIN(S1),120+72
*COS(S1))
1060 NEXT S2 :: S=0 :: S1=PI/2
1070 M1=M1-PI/30
1080 CALL LOCATE(#17,88-64*SIN(M1),120+64
*COS(M1))
1090 M=M+1
```

```
1100 IF M=60 THEN M=0 ELSE 870
1110 REM COULEUR DES CHIFFRES
1120 CALL SOUND(100,400,0,4000,2):: M1=PI/2 :: A=A+1
1130 IF H<9 THEN B=A ELSE B=A+3
1140 CALL COLOR(#A-1,2):: IF H>9 THEN CALL COLOR(#B-1,2)
1150 IF A=13 THEN A,B=1
1160 IF A=12 AND X=1 THEN JR=JR+1 :: X=0 :: SA=SA+1
1170 IF SA=8 THEN SA=1
1180 JOUR$=SEG$(J$,8*SA-7,8)
1190 CALL COLOR(#A,5):: CALL COLOR(#B,5)
1200 REM CALENDRIER
1210 IF (MO=1 OR MO=3 OR MO=5 OR MO=7 OR MO=8 OR MO=10)AND JR=32 THEN MO=MO+1 :: JR=1 :: GOTO 1250
1220 IF (MO=4 OR MO=6 OR MO=9 OR MO=11)AND JR=31 THEN MO=MO+1 :: JR=1 :: GOTO 1250
1230 IF MO=2 AND JR=FEV THEN MO=MO+1 :: JR=1 :: GOTO 1250
1240 IF MO=12 AND JR=32 THEN MO=1 :: JR=1 :: AN=AN+1
1250 MOI$=SEG$(M$,9*MO-8,9):: GOTO 870
1260 END
1270 ! CALCUL DU JOUR ET DU MOIS
1280 J$=" lundi   mardi  mercredi jeudi  vendredi samedi dimanche"
1290 M$=" janvier  fevrier   mars    avril     mai     juin    juillet   aout   septembre octobre novembre decembre "
1300 A1=INT(AN/100):: A2=AN-100*A1 :: N=0
1310 IF MO>2 THEN 1400
1320 N=2
1330 IF A2=0 THEN 1380
1340 R=A2-4*INT(A2/4)
1350 IF R<>0 THEN 1400
1360 N=1
1370 GOTO 1400
1380 R=A1-4*INT(A1/4)
1390 IF R=0 THEN N=1
1400 CC=INT(365.25*A2)+INT(30.56*MO)+N+JR
1410 SA=3+CC-7*INT((CC+2)/7)
1420 JOUR$=SEG$(J$,8*SA-7,8)
1430 MOI$=SEG$(M$,9*MO-8,9)
1440 RETURN
```

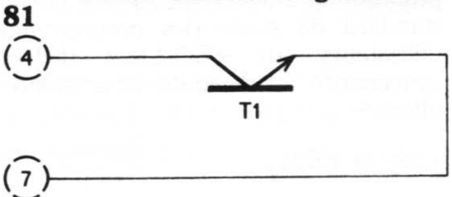
Un détecteur de lumière pour le TI-99

Guy Fix

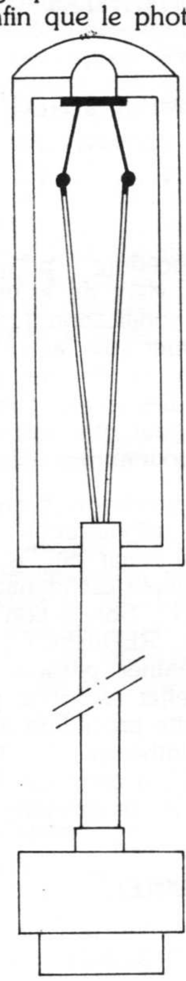
Ce détecteur utilise l'entrée des manettes de jeu. Il est possible de le brancher sur la manette A ou B ou, pourquoi pas, sur les deux entrées.
Le brochage et le branchement du détecteur sont donnés pour un déclenchement simulant le bouton de tir des manettes.

Manette A - 7 et 4
Manette B - 2 et 4

T1 - Photo-transistor genre TIL 81



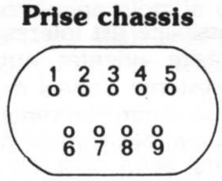
Le montage peut être effectué dans un tube, afin que le photo-transistor ne soit pas affecté par la lumière ambiante.



- Prise CANON 9 points ou SOURIAU DEF-9S

Pour toutes les autres utilisations, le brochage de la prise est donné ci-dessous.

Prise chassis



1 - libre
2 - manette B
3 - haut
4 - bouton de tir
5 - gauche
6 - libre
7 - manette A
8 - bas
9 - droite

Exemple : pour un branchement simulant le déplacement de la manette B vers la droite, raccorder le détecteur sur 2 et 9.

Explication sommaire du fonctionnement

Si T1 est éclairé, il se sature et "relie" les points 4 et 7, modifiant ainsi l'état de la variable affectée au bouton de tir (code 18).

NDLR : cette petite réalisation constitue une première étape vers la réalisation d'un crayon optique. En fait, toute la partie "hardware" est résumée ci-dessus. Le reste est uniquement un problème de logiciel. En résumé : il faut écrire une routine en Assembleur (le Basic est beaucoup trop lent pour ce type d'application) capable de détecter la position du crayon par rapport à l'écran.
Philippe Massénat nous a aimablement proposé sa collaboration pour effectuer la collecte des informations concernant ce sujet. Si vous avez déjà travaillé sur le sujet, ou si vous avez trouvé la solution du problème, vous pouvez écrire à : Philippe Massénat - 33, Avenue Habert de Montmort - 78320 Le Mesnil-Saint-Denis. ■

Le club M.T.I. se présente

Nous avons le plaisir de vous faire connaître l'existence à Nice d'un club d'utilisateurs du TI-99/4A.

Ce club existe depuis le mois de juillet 1983 et regroupe des adhérents à travers toute la France. Il est organisé de façon à pouvoir fonctionner par correspondance. Les adhérents reçoivent un bulletin périodique, ainsi que deux programmes gratuits à choisir chaque trimestre dans une liste de programmes éditée dans le bulletin (seul le support cassette ou disquette est facturé). Une permanence téléphonique, réservée aux adhérents, est assurée de 8 heures à 12 heures, et de 14 heures à 18 heures.

Des accords passés avec des fournisseurs permettent aux adhérents de se procurer du matériel à prix réduit.

Le club est constitué sous la forme d'une association à but non lucratif. L'adresse est :
M.T.I. - 6, rue Georges Ville - 06300 NICE - Téléphone : (93) 26 48 48 ■

Le système P-UCSD

Gérard Santraille

La carte P-Code permet d'accéder au système P-UCSD, parfois appelé POS (Pascal Operating System). Ce système d'exploitation, même s'il est assez lourd, possède des capacités intéressantes.
Revenons très brièvement sur les commandes-système primaires du POS, qui apparaissent sur la ligne guide, lorsque le système est démarré.

- "I(nit" ré-initialise le système, et lance l'exécution de "SYSTEM.STARTUP", si ce fichier est présent sur le volume racine (nous reviendrons sur cette possibilité).
- "E(dit" charge l'éditeur (appelé "SYSTEM.EDITOR") et le fichier de travail, s'il existe (SYSTEM.WRK).
- "R(un" exécute le fichier de travail. Si seul "SYSTEM.WRK.TEXT" est présent sur la disquette ou si le fichier-code est une version antérieure du fichier-texte, le compilateur est appelé automatiquement. De même, si le fichier-code nécessite une édition de liens, le "linker" est chargé.
- "C(omp" charge le compilateur (appelé "SYSTEM.COMPILER") et opère directement sur le fichier de travail (s'il existe).
- "L(ink" charge l'éditeur de liens (appelé "SYSTEM.LINKER") et agit automatiquement sur le fichier de travail.
- "X(ecute" lance l'exécution d'un fichier-code. Cette commande permet également de spécifier certaines options modifiant l'environnement du système. On peut, par exemple, déclarer directement le préfixe par défaut, sans faire appel au "FILER". On peut aussi, et c'est vraisemblablement l'une des particularités les plus intéressantes du POS, réorienter les fichiers standards d'entrée/sortie du système ou d'un programme. Nous reviendrons largement sur cette option, car elle conditionne entièrement l'utilisation des fichiers moniteur.
- "A(ssem" charge l'assembleur et opère directement sur le fichier de travail s'il est présent.
- "F(ile" appelle le "FILER".
- "H(alt" est une commande supportée par la version IV.0, qui permet de sortir du P-SYSTEME; on revient ainsi à la mire d'introduction.
- "U(ser restart" permet de relancer directement le dernier programme exécuté, sans modifier les valeurs des différentes variables.
- "M(onitor" permet de débuter, de suspendre ou de terminer une session moniteur.

Intérêt et utilisation de "SYSTEM.STARTUP"

Lorsque le système est démarré ou ré-initialisé, le fichier "SYSTEM.STARTUP" est automatiquement exécuté s'il est présent sur le volume racine.
"SYSTEM.STARTUP" peut être un fichier-code quelconque.

Cette possibilité est intéressante lorsqu'on désire orienter automatiquement le système (travail à distance à partir d'un autre terminal, en dirigeant les entrées/sorties du système sur REMIN: et REMOUT:) ou initialiser correctement une imprimante (par exemple, en exécutant l'utilitaire MODRS232) dès la mise en service, ou encore diriger les entrées du système sur un fichier moniteur. Pour ma part, sur la disquette contenant l'éditeur, j'ai toujours sous le nom de "SYSTEM.STARTUP" une version modifiée et automatique de l'utilitaire "MODRS232" qui initialise correctement mon imprimante (PIO).

Modification des fichiers d'entrée/sortie des programmes ou du système

Modification des entrées/sorties des programmes

Les programmes Pascal admettent deux fichiers standards (INPUT et OUTPUT) qu'il n'est pas nécessaire de définir. "INPUT" correspond au clavier et "OUTPUT" à l'écran. Il est possible de changer ces fichiers par défaut par la clause "PI=nom de fichier" (pour Program Input) et "PO=nom de fichier" (pour Program Output), accessible à partir de X(ecute du mode commande. Ainsi, presser X en mode commande provoque l'affichage de :

Xecute
what file?

Si on répond par "PO=printer:", toutes les impressions des programmes se feront, sauf spécification contraire, sur l'imprimante. Il est possible d'effectuer plusieurs modifications à la fois, en les séparant par un espace. Ainsi, "PO=printer: PI=remin:" affecte les fichiers d'entrée et de sortie des programmes utilisateurs.

On peut également modifier ces fichiers standards par un programme utilisateur, au moyen de la procédure "CHAIN", qui a le même effet que Xecute. La procédure "CHAIN" permet, comme son nom l'indique, d'enchaîner un programme à un autre (il s'agit en fait d'un overlay dynamique semblable au RUN "programme" du Basic Etendu). L'action de cette procédure n'est pas immédiate : la chaîne d'options d'exécution spécifiée est mise en attente et ne sera effectivement exécutée qu'à la fin du programme principal. Le programme suivant modifie le fichier standard de sortie des programmes utilisateurs sur #5:list.text (fichier sauvegardé sur disquette pour édition ultérieure).

```
Program ESSAI;

var .................
    .................

begin
  chain('PO=£5:LIST.TEXT');
  ......................
  ......................
end;
```

Cette procédure permet donc, comme on vient de le voir, d'effectuer une modification (gardons toujours à l'esprit que l'action n'est exécutée qu'à la fin du programme appelant). De façon plus générale, "CHAIN" peut être utilisé pour lier plusieurs programmes utilisateurs.

Une autre procédure permet d'effectuer une modification par programme : il s'agit de "REDIRECT", dont la syntaxe est identique à celle de "CHAIN". Son action est immédiate, mais "REDIRECT" ne permet pas d'enchaîner plusieurs programmes; son effet est donc plus limité. Comme cette procédure appartient à la bibliothèque "COMMANDIO.CODE", il convient de la déclarer dans le programme appelant. Ainsi :

```
Program EXEMPLE;

Uses COMMANDIO.CODE;
```

```
begin
redirect('PO=PRINTER:');
........................
........................
end
```

écrira sur l'imprimante chaque fois qu'il rencontrera un ordre d'impression ne désignant pas nommément le fichier destinataire. De la même façon, on peut spécifier le fichier d'entrée par défaut du système au moyen de "PI= nomdefichier".

Modifications des entrées/sorties du système

A la mise sous tension, lorsqu'on accède "naturellement" au P-SYSTEM, l'ordinateur communique avec l'utilisateur par le clavier et l'écran (ou le moniteur pour les mieux équipés), rien de plus normal jusque là. Il est cependant possible d'orienter le système sur l'interface RS232, sur l'imprimante, ou même sur un fichier d'une disquette. Cette modification s'effectue de la même façon que précédemment, c'est-à-dire au moyen des procédures "CHAIN" et "REDIRECT".

La syntaxe est :

CHAIN('I=REMIN: O=REMOUT:')

avec toujours I pour INPUT et O pour OUTPUT.
Comme pour les modifications des entrées/sorties des programmes, on peut spécifier plusieurs options dans le même appel à la procédure, en les séparant par un espace. Par exemple, une orientation des entrées du système sur un modem connecté à l'ordinateur, via l'interface RS232, permet de commander votre système à distance (ce n'est, hélas, pas tout à fait aussi simple que cela). De la même façon, si l'on spécifie l'imprimante comme fichier de sortie du système, tous les messages-guide seront imprimés et n'apparaîtront plus à l'écran.

Une possibilité intéressante consiste à diriger les entrées du système vers un fichier sur disquette. Ce fichier de type "TEXT" ou "DATA", qui peut être créé par l'éditeur de texte ou à partir d'une session moniteur, est alors lu séquentiellement par le système, et exécuté au fur et à mesure. Supposons que le fichier "DEMO.TEXT" soit situé sur le volume "GG1:", et qu'il contienne le texte suivant :

```
FD19
NYQE*LETTRE
/R/Monsieur//Madame/ jb
```

Si, en mode commande, on presse "X" et que l'on réponde "I=GG1:DEMO.TEXT" quand apparaît le prompteur "What File?", le système réagit comme si l'utilisateur pressait une à une les touches contenues dans le fichier "GG1:DEMO.TEXT".

Le "FILER" est chargé (F) et la date mise au 19 du mois courant (D19 puis passage à la ligne interprété comme un RETURN), le fichier de travail est ensuite détruit (N) et le Directory remis à jour (Y en réponse à Workfile removed update directory ?). Le système quitte ensuite le FILER (Q) et charge l'éditeur (E), qui charge à son tour le fichier "LETTRE.TEXT" du volume racine (*LETTRE). Rappelons que le volume racine (identifié par *) est le volume présent en "#4:", lorsqu'on accède au P-SYSTEM. Le système remplace ensuite toutes les chaînes 'Monsieur' par 'Madame' (/R/Monsieur//Madame/). L'espace entre Madame/ et jb permet de poursuivre l'exécution si la chaîne "Monsieur" n'a pas été trouvée (ERROR: Pattern not found. type<SP>); le système repositionne alors le curseur au début du texte (JB), puis rend la main à l'utilisateur (fin de fichier).

Comme on vient de le voir, le système peut effectuer automatiquement (et séquentiellement) toute une série de commandes et/ou de manipulations, à condition d'être dirigé vers un fichier adéquat. On peut créer un fichier moniteur de deux façons différentes :

1) à partir de l'éditeur de texte. D'un emploi très souple, le fichier obtenu pourra évidemment être réédité et corrigé. Le seul petit inconvénient de ce procédé est qu'un fichier créé par l'éditeur occupe au minimum 4 blocs (il mémorise en effet plusieurs autres informations qui ne nous intéressent pas) alors qu'un fichier moniteur d'un seul bloc est généralement suffisant;
2) en mémorisant une session moniteur. En mode commande, la pression sur M provoque l'affichage du prompteur. Monitor: B(egin E(nd A(bort puis S(uspend R(esume, chaque fonction permettant respectivement de débuter, finir, annuler, suspendre et reprendre une session moniteur.

Remarque : lors d'une session moniteur, il n'est pas possible de mémoriser la pression de M (qui permet justement d'avoir accès au moniteur).

Modification des entrées (système ou programmes) par chaîne (string)

Nous avons vu que lors d'une modification, qu'elle soit manuelle par le "X(ecute" du mode commande, ou automatique à partir d'un programme utilisateur par les procédures "CHAIN" ou "REDIRECT", on spécifiait toujours "I=nomdefichier" ou "PI=nomdefichier". Il est également possible d'écrire "I=chaîne" ou "PI=chaîne". Le système ou le programme s'oriente alors sur le CONTENU de la chaîne.

Ainsi, I="FD19,NYQ" charge le "FILER", met la date au 19 du mois courant, efface le fichier travail, met à jour le directoire, et retourne au mode commande. Cette écriture plus légère permet de ne pas accéder systématiquement à un fichier, lorsque les manipulations à effectuer ne sont pas trop nombreuses.
Remarque : dans une chaîne utilisée pour orienter les entrées, les virgules sont interprétées comme des retours-chariot.

Exemples d'application

Mise à jour automatique de la date à chaque session du P-SYSTEM

Si vous placez le programme suivant sur la disquette contenant l'éditeur, sous le nom "SYSTEM.STARTUP", chaque fois que l'on accède au P-system ou que l'on presse I en mode commande, le système demandera la date (à condition, bien entendu, que la disquette soit présente en #4).
Remarque : pressez simplement "ENTER" pour ne pas modifier la date.

```
Program DATE;

var action,date:string;


begin
write(chr(12),'date : ',chr(7));
readln(date);
if date=''
then
 exit(program)
else
 begin
  action:=concat('I="FD',date,',"');
  redirect(action)
 end
end.
```

Conversion automatique de caractères spéciaux

Le TI-99 ne possédant pas de caractères accentués, il est nécessaire de faire appel à un "traducteur" qui interprétera certains caractères (définis à l'avance par l'utilisateur), lors d'une édition sur imprimante. Le programme suivant convertit automatiquement un fichier de type "TEXT", comportant certains caractères spéciaux, en caractères accentués ou en commandes de l'imprimante (chacun devra l'adapter à sa configuration : il

s'agit ici d'une SEIKOSHA GP100). Rappelons au passage que l'éditeur de texte n'accepte pas de caractère de contrôle (ils sont systématiquement remplacés par un point d'interrogation), et qu'il est donc impossible de commander l'imprimante directement à partir d'un fichier "TEXT", lors de son transfert.

```
Program impression;

var
      imp,lect:interactive;
      car:char;
      asc:integer;
      fichier:string;

begin
  write(chr(7),' fichier : ');
  readln(fichier);
  fichier:=concat(fichier,'.text');
  reset(lect,fichier);
  rewrite(imp,'PRINTER:');
  while not eof(lect)do
    begin
      read(lect,car);
      if eoln(lect)
        then
          writeln(imp)
        else
          begin
            asc:=ord(car);
            case car of
              '{':
                  asc:=14;
              {double largeur}
              '}':
                  asc:=15;
              {simple largeur}
              '[':
                  asc:=166;
              {e accent aigu}
              ']':
                  asc:=167;
              {e accent grave}
              '~':
                  asc:=168;
              {e accent circonflexe}
              '@':
                  asc:=162;
              {a accent aigu}
              '\':
                  asc:=182;
              {u accent aigu}
              '|':
                  asc:=183;
              {u accent circonflexe}
              '^':
                  asc:=179;
              {o accent circonflexe}
            end;
            write(imp,chr(asc))
          end;
    end;
  writeln(imp)
end.
```

Ce programme est aisément adaptable ou extensible selon les besoins. Il suffit de modifier ou d'allonger la liste du :

```
case car of
...........
...........
end;
```

Lorqu'on désire visualiser le contenu du fichier à l'écran, il convient de redonner aux caractères spéciaux une forme plus sympathique, ou surtout plus intelligible. Deux façons de procéder :

1) On écrit un petit programme qui explore tout le fichier "TEXT", et qui remplace au fur et à mesure (lecture séquentielle) tous les caractères codés.
2) On fait exécuter le même travail par l'éditeur de texte, qui comporte une commande parfaitement adaptée à ce type de manipulation.

Ainsi, le programme suivant redirige les entrées du système sur une chaîne. Le système charge alors l'éditeur qui, au moyen de "REPLACE", effectue le travail demandé.

```
program DECODE;

(*$U #4:COMMANDIO.CODE *)
uses commandio;

const
      cght=',/r/[//e/ jb/r/]//e/ jb/r/~//e/ jb/r/@//a/ jb/r/`/';
      cght2='/u/ jb/r/^//o/ jb/r/{/// jb/r/}/// jb/r/|//u/ jb"';

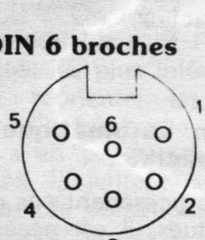
var
      fichier:string;

begin
  write(chr(7),' Nom du fichier : ');
  readln(fichier);
  chain(concat('I="E',fichier,'.TEXT',cght,cght2))
end.
```

Comme on peut le constater le POS recèle d'étonnantes possibilités. La manipulation des fichiers moniteurs et les modifications de ses entrées-sorties ne sont que deux des innombrables facultés de celui-ci. Le Pascal UCSD, quant à lui, possède également au niveau de sa syntaxe propre de nombreuses possibilités dont nous parlerons dans de prochains articles. ■

Brochage de la prise vidéo du TI-99

Fiche DIN 6 broches



1 - + 12 Volts
2 - Vidéo composite - standard NSTC
3 - R-Y
4 - B-Y
5 - Sortie audio
6 - Masse
R-Y et B-Y sont les signaux nécessaires à l'interface R.V.B. ■

Librairie Pascal

Lors de l'utilisation, en Pascal, de procédures ou de fonctions propres au TI-99 (manipulation des lutins et des couleurs, effets sonores, définitions de caractères, utilisation du synthétiseur de paroles), il est nécessaire de déclarer la librairie que l'on utilise. S'il s'agit de la librairie standard écrite par Texas Instruments (9 bibliothèques regroupées par "SYSTEM.LIBRARY") ne pas oublier, lors de la compilation, de déclarer le volume contenant ce fichier. En effet, "SYSTEM.LIBRARY" est pris par défaut, à condition qu'il se trouve en #4, lorsque le compilateur rencontre "USES BIBLIOTHEQUE". Si ce n'est pas le cas, il faut alors écrire :

(*$u NOMDUVOLUME:SYSTEM.LIBRARY *) ■

Un mode "test" sur le module Alpiner

Dans le numéro 3 de "99 Magazine", nous vous avions donoeé la méthode pour accéder au mode test de "Munch Man". Avec le module de jeu "Alpiner", il est possible de faire la même chose. Rappelons que la séquence : *#* (SHIFT-8, SHIFT-3 et SHIFT-8) permet l'entrée dans ce mode particulier.

Les particularités du Basic Etendu
Les Lutins (Sprites)

Julien Thomas

Le TI-99/4A est doté, pour la gestion de l'écran, d'un microprocesseur spécialisé (TMS 9918A) qui permet la création et l'animation de lutins. Pour le dictionnaire, les lutins sont des "petits génies espiègles, généralement bienveillants, parfois dangereux". Pour le TI-99, ce sont des objets auxquels nous pouvons donner certains paramètres, et qui apparaissent sur l'écran en superposition de l'affichage matriciel classique. Le module "Basic Etendu" comporte des sous-programme résidents (accessibles dès la mise sous tension) qui agissent directement sur les lutins.

Dans cet article, vous trouverez de nombreux petits programmes. Afin de faciliter leur chargement à partir de la cassette d'accompagnement, nous avons décidé de les regrouper dans un unique programme. Pour faire "tourner" ces exemples, vous devrez donc utiliser la commande "RUN numéro de ligne". Par exemple, pour CALL MAGNIFY, vous devrez taper : RUN 920. Pour sortir des programmes, il faudra utiliser "CLEAR" (FCTN-4).

CALL SPRITE

CALL SPRITE est l'instruction de base du système. Elle permet de créer un ou plusieurs lutins en leurs affectant une forme, une couleur, une position, etc... Sa syntaxe est :

- CALL SPRITE(#numéro du lutin,code de caractère,couleur,numéro de ligne,numéro de colonne[,vitesse verticale,vitesse horizontale]],autres lutins])

Les paramètres placés entre crochets sont facultatifs.

Le TI-99/4A peut générer jusqu'à 32 lutins, mais seulement 28 sont autorisés en Basic Etendu. Le numéro du lutin doit donc être un entier compris entre 1 et 28 inclus.

Le code de caractère doit être un entier compris entre 32 et 143, et correspond à un caractère qu'il est possible de redéfinir avec l'instruction CALL CHAR, tout comme en Basic TI.

Les couleurs sont les mêmes qu'en Basic TI (de 1 pour incolore à 16 pour blanc). Seule la couleur du lutin (points allumés) est modifiable, la couleur de fond (points éteints) étant toujours 1 (incolore).

Les numéros de ligne et de colonne déterminent la position du lutin sur l'écran lors de sa création, et correspondent à l'endroit où le coin supérieur gauche du lutins se trouvera.

Exemple :

```
100 ! *******************
110 ! *  CALL  SPRITE  *
120 ! *Un seul lutin fixe*
130 ! *******************
140 !
150 CALL CLEAR
160 CALL SPRITE(#1,65,7,94,124)
170 GOTO 170
```

Le moins que l'on puisse dire est que cet exemple n'est pas très impressionnant. Cependant, il démontre qu'il est possible de placer un lutin à n'importe quel endroit. Christian Monfort, dans son programme "Horloge", a très astucieusement utilisé cette possibilité. Puisque nous avons utilisé le code 65 pour créer notre lutin, il a la forme d'un "A".

Autre exemple :

```
180 ! *******************
190 ! *   CALL  SPRITE  *
200 ! * Un seul lutin en *
210 ! *    mouvement    *
220 ! *******************
230 !
240 CALL CLEAR
250 CALL SPRITE(#1,65,7,94,124,1
0,-10)
260 GOTO 260
```

Cette fois, le lutin se déplace et, ce qui est remarquable, sans qu'un programme ne soit obligé d'intervenir. Les vitesses de ligne et de colonne doivent être comprises entre −128 et 127. La valeur 0 donne un état fixe. Pour la vitesse de colonne, une valeur négative provoque un déplacement de droite à gauche, alors qu'une valeur positive provoque un déplacement de gauche à droite. Pour la vitesse de ligne, une valeur négative donne un déplacement du bas vers le haut, alors qu'une vitesse positive provoque un déplacement de haut en bas. En jouant sur les vitesses de ligne et de colonne, il est possible d'obtenir des déplacements dans n'importe quelle direction.

```
270 ! *******************
280 ! *   CALL  SPRITE  *
290 ! *  Deux lutins en  *
300 ! *    mouvement    *
310 ! *******************
320 !
330 CALL CLEAR
340 CALL SPRITE(#1,65,7,94,124,1
0,-10,#2,66,13,94,124,10,10)
350 GOTO 350
```

Cette fois, nous avons deux lutins se déplaçant dans des directions opposées. En fait, seule la place disponible dans une ligne de programme limite le nombre de lutins qu'il est possible de créer dans une seule instruction CALL SPRITE.

CALL COLOR

Lorsque l'on a fait apparaître un lutin avec CALL SPRITE, il peut être intéressant de changer sa couleur. C'est ce que fait CALL COLOR, qui s'utilise de la manière suivante :

- CALL COLOR(#numéro de lutin,couleur[,autre lutin])

Cette instruction n'est pas la même que le CALL COLOR pour modifier la couleur d'un groupe de caractères. Si vous essayez de mélanger les caractères et les lutins dans cette instruction, vous obtiendrez un splendide message d'erreur. Dans l'exemple suivant, nos deux lutins changent continuellement de couleur de façon aléatoire.

```
360 ! *******************
370 ! *    CALL COLOR    *
380 ! *******************
390 !
400 CALL CLEAR
410 CALL SCREEN(1)
420 CALL SPRITE(#1,65,7,94,124,1
0,-10,#2,66,13,94,124,10,10)
430 RANDOMIZE :: CALL COLOR(#1,I
NT(RND*14)+3,#2,INT(RND*14)+3)::
 GOTO 430
```

CALL CHAR

Jusqu'à présent, nous nous étions contentés, pour nos deux lutins, de caractères prédéfinis. L'instruction CALL CHAR autorise un "remodelage" des caractères et, si elle s'utilise de la même façon qu'en Basic TI, elle comporte néanmoins quelques différences. En Basic TI, un CALL CHAR ne peut redéfinir qu'un seul caractère. Ici, on peut redéfinir plusieurs caractères simultanément et une chaîne (codage hexadécimal de la forme) peut contenir jusqu'à 64 éléments :

- si la chaîne contient moins de 16 éléments, un seul caractère est redéfini;
- si elle contient entre 17 et 32 éléments, deux caractères sont redéfinis;
- si elle contient entre 33 et 48 éléments, trois caractères;
- si elle contient entre 49 et 64 éléments, quatre caractères.

Dans tous les cas, les éléments manquants sont remplacés par des zéros. Si une chaîne contient plus de 64 éléments, le surplus est ignoré. Lorsque plusieurs caractères sont définis simultanément, l'ordre suivant est respecté :

X	X+2
X+1	X+3

Le petit segment de programme suivant tranforme les lutins des exemples précédents en un rectangle et un losange :

```
440 ! ********************
450 ! *    CALL  CHAR    *
460 ! ********************
470 !
480 CALL CLEAR
490 CALL SCREEN(1)
500 CALL CHAR(65,"FF8181818181818181
FF183C66C3C3663C18")
510 CALL SPRITE(#1,65,7,94,124,1
0,-10,#2,66,13,94,124,10,10)
520 RANDOMIZE :: CALL COLOR(#1,I
NT(RND*14)+3,#2,INT(RND*14)+3)::
 GOTO 520
```

CALL MOTION

Cette instruction permet de modifier le déplacement d'un ou de plusieurs lutins, ou de mettre en mouvement un ou plusieurs lutins immobiles. La syntaxe est très simple :

- CALL MOTION(#numéro de lutin,vitesse de ligne,vitesse de colonne[,autre lutin])

Dans le premier exemple, les vitesses sont choisies aléatoirement. Notons au passage que INT(RND*20)*SGN(RND−.5) retourne une valeur comprise entre −19 et 19 puisque SGN(RND−.5) retourne −1, 0 ou 1.

```
530 ! ********************
540 ! *   CALL  MOTION   *
550 ! * premier  exemple *
560 ! ********************
570 !
580 CALL CLEAR
590 CALL SPRITE(#1,65,7,94,124,1
0,-10,#2,66,13,94,124,10,10)
600 RANDOMIZE :: CALL MOTION(#1,
INT(RND*20)*SGN(RND-.5),INT(RND*
20)*SGN(RND-.5),
#2,INT(RND*20)*SGN(RND-.5),INT(R
ND*20)*SGN(RND-.5)):: GOTO 600
```

Le deuxième exemple autorise un déplacement contrôlé. Vous devez appuyer sur les touches fléchées pour obtenir le déplacement du lutin.

```
610 ! ********************
620 ! *   CALL  MOTION   *
630 ! * deuxieme exemple *
640 ! ********************
650 !
660 CALL CLEAR
670 CALL SCREEN(1)
680 CALL CHAR(32,"80808080808080
FFFF818181818181FF")
690 CALL COLOR(1,4,8)
700 CALL SPRITE(#1,33,7,94,124)
710 CALL KEY(0,K,S):: IF S=0 THE
N X=0 :: Y=0 :: GOTO 760
720 IF K=69 THEN X=-10 :: Y=0 ::
 GOTO 760
730 IF K=83 THEN X=0 :: Y=-10 ::
 GOTO 760
740 IF K=68 THEN X=0 :: Y=10 ::
GOTO 760
750 IF K=88 THEN X=10 :: Y=0 ELS
E 710
760 CALL MOTION(#1,X,Y):: GOTO 7
10
```

CALL POSITION

Cette instruction vous permet de savoir où se trouvent les lutins. Les positions verticales et horizontales sont retournées dans des variables. CALL POSITION s'utilise ainsi :

- CALL POSITION(#numéro du lutin,variable ligne,variable colonne[,autre lutin])

Le petit exemple placé ci-dessous met deux lutins en mouvement et reporte en permanence leurs positions. USING, associé à DISPLAY autorise un affichage "formaté" :

```
770 ! ********************
780 ! *  CALL  POSITION  *
790 ! ********************
800 !
810 CALL CLEAR
820 DISPLAY AT(21,2):"1er LUTIN
     2eme LUTIN" :: DISPLAY AT(
23,3):"Ligne:
      Ligne:" :: DISPLAY AT(24,
1):"Colonne:        Colonne:"
830 DISPLAY AT(22,2):"---------
     ---------"
840 CALL SPRITE(#1,65,7,94,124,1
0,-10,#2,66,13,94,124,10,10)
850 RANDOMIZE :: CALL MOTION(#1,
INT(RND*20)*SGN(RND-.5),INT(RND*
20)*SGN(RND-.5),
#2,INT(RND*20)-SGN(RND-.5),INT(R
ND*20)*SGN(RND-.5))
860 CALL POSITION(#1,X1,Y1,#2,X2
,Y2):: DISPLAY AT(23,10)SIZE(3):
USING "###":X1 :
: DISPLAY AT(24,10)SIZE(3):USING
 "###":Y1
870 DISPLAY AT(23,26):USING "###
":X2 :: DISPLAY AT(24,26):USING
"###":Y2 :: GOTO
 850
```

CALL MAGNIFY

C'est l'instruction qui possède la syntaxe la plus simple :

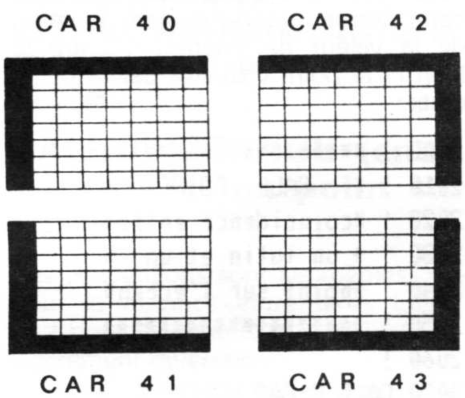
- CALL MAGNIFY(valeur)

La valeur doit être un entier de 1 à 4. Examinons les effets de cette instruction :

1 - est la valeur par défaut, donc celle que nous utilisons depuis le début de cet article. Le lutin utilise un seul code de caractère et occupe la même surface qu'un caractère standard;
2 - le lutin utilise toujours un seul code de caractère mais, cette fois, chaque point définissant le lutin occupe quatre points d'écran (pixels), ce qui fait que l'on couvre maintenant une surface identique à celle nécessaire à la représentation de quatre caractères standards;
3 - cette fois, comme pour CALL MAGNIFY(1), les points du lutin occupent un point sur l'écran. Par contre, quatre codes de caractères sont utilisés. Le premier caractère est celui qui représente le quart supérieur gauche du lutin et son code doit être divisible par quatre. Si tel n'est pas le cas, le premier caractère utilisé sera le plus proche à posséder un code divisible par quatre, en al-

lant vers les nombres les plus forts;
4 - il s'agit là de la combinaison des valeurs 2 et 3. En effet, les lutins utilisent quatre codes de caractères et chacun de leurs points occupe quatre points sur l'écran. Chaque lutin utilise une surface équivalente à celle nécessaire à la représentation de seize caractères standards.

Le petit programme suivant visualise l'action de CALL MAGNIFY. Les caractères utilisés sont redéfinis ainsi :



Les touches 1, 2, 3 et 4 retournent un "CALL MAGNIFY" correspondant :

```
880 ! ********************
890 ! *   CALL MAGNIFY   *
900 ! ********************
910 !
920 CALL CLEAR
930 CALL CHAR(40,"FF808080808080
808080808080808080FFFF010101010101
0101010101010101
FF")
940 CALL SPRITE(#1,40,7,94,124)
950 CALL KEY(0,K,S):: IF (K<49)O
R(K>52)THEN 950 ELSE CALL MAGNIF
Y(K-48):: GOTO 9
50
```

CALL PATTERN

Cette instruction est utilisée pour modifier l'apparence d'un ou de plusieurs lutins. Ceci est très utile pour animer des objets ou des personnages. La modification du lutin est effectuée en lui affectant un code de caractère différent à chaque utilisation de CALL PATTERN. Syntaxe de l'instruction :

- CALL PATTERN(#numéro du lutin,code de caractère[,autre lutin])

Le premier exemple proposer pour illustrer cette instruction anime un objet volant parfaitement identifié, puisqu'il s'agit de la soucoupe du jeu "TI-Invaders" :

```
960 ! *******************
970 ! *   CALL PATTERN   *
980 ! * Premier exemple *
990 ! *******************
1000 !
1010 CALL CLEAR
1020 CALL SCREEN(1)
1030 CALL MAGNIFY(3)
1040 DATA 01033FFF77FF3F07,0,80C
0FCFF77FFFCE,0,01033FFFBBFF3F07,
0,80C0FCFFBBFFFC
E,0
1050 DATA 01033FFFDDFF3F07,0,80C
0FCFFDDFFFCE,0,01033FFFEEFF3F07,
0,80C0FCFFEEFFFC
E,0
1060 FOR BD=40 TO 55 :: READ C$
:: CALL CHAR(BD,C$):: NEXT BD
1070 CALL SPRITE(#1,40,11,90,10,
0,9)
1080 FOR BP=40 TO 52 STEP 4 :: C
ALL PATTERN(#1,BP):: FOR D=0 TO
5 :: NEXT D :: N
EXT BP :: GOTO 1080
```

Le second exemple représente un petit personnage en marche et, s'il ressemble à "E.T.", ce n'est cette fois qu'un pur hasard :

```
1090 ! ******************
1100 ! * CALL  PATTERN  *
1110 ! *Deuxieme exemple*
1120 ! ******************
1130 !
1140 CALL CLEAR
1150 CALL SCREEN(13)
1160 CALL CHAR(40,"1C1C1038385C5
C9A9A18181010103018000000000000
0000")
1170 CALL CHAR(44,"1C1C103838385
C5C5C18141224242630000000000000
0000")
1180 CALL CHAR(48,"1C1C103838383
8383818182824446406000000000000
0000")
1190 CALL CHAR(52,"1C1C103838385
C5C5C181818284828 0C000000000000
0000")
1200 CALL COLOR(1,5,5):: CALL MA
GNIFY(3):: CALL SPRITE(#1,40,16,
177,1):: X=1
1210 X=1
1220 FOR B=40 TO 52 STEP 4 :: CA
LL PATTERN(#1,B)
1230 X=X+1 :: IF X=256 THEN 1210
1240 CALL LOCATE(#1,177,X)
1250 FOR D=0 TO 1 :: NEXT D
1260 NEXT B
1270 GOTO 1220
```

CALL LOCATE

Il est possible, grâce à cette instruction, de placer un lutin à n'importe quel endroit sur l'écran. Une instruction CALL LOCATE est composée comme suit :

- CALL LOCATE(#numéro du lutin,numéro de ligne,numéro de colonne[,autre lutin])

Dans le programme suivant, un lutin est créé au milieu de l'écran et, au bout de quelques secondes, il se met à se déplacer suivant un parcours elliptique :

```
1280 ! ******************
1290 ! *   CALL LOCATE   *
1300 ! ******************
1310 !
1320 CALL CLEAR
1330 CALL CHAR(32,"000000FFFF000
0003C7EFFFFFFFF7E3C")
1340 CALL COLOR(1,5,8)
1350 CALL SCREEN(1)
1360 B1=PI/180 :: B2=B1*360 :: B
3=B1*3
1370 CALL SPRITE(#1,33,7,94,124)
:: FOR BD=0 TO 500 :: NEXT BD
1380 FOR B=B1 TO B2 STEP B3 :: C
ALL LOCATE(#1,98+40*SIN(B),128+9
0*COS(B)):: NEXT
 B :: GOTO 1380
```

CALL COINC(ALL,variable)

Ceci autorise la détection d'une éventuelle coïncidence entre deux lutins. Dans le cas de l'utilisation du paramètre "ALL", le chevauchement de n'importe quel lutin avec n'importe quelle autre lutin retoune la valeur −1 dans la variable utilisée. Si aucune coïncidence n'est détectée, la variable contient 0.

Le premier exemple met en évidence que, lorsqu'on utilise "ALL", pour qu'une coïncidence soit détectée, il faut une superposition physique. Les coordonnées des lutins sont les mêmes pour les quatre lutins mais, comme les lutins ne se chevauchent pas, la variable "C" contient 0.

```
1390 ! ******************
1400 ! *   CALL COINC   *
1410 ! * (ALL, variable) *
1420 ! * premier exemple *
1430 ! ******************
1440 !
1450 CALL CLEAR
1460 CALL COLOR(1,16,8)
1470 CALL SCREEN(4)
1480 CALL CHAR(32,"1824428181422
418")
1490 CALL MAGNIFY(4)
```

```
1500 DATA FF8080808080808,808080
80808080FF,FF01010101010101,0101
0101010101FF
1510 DATA 007F40404040404,404040
4040407F,00FE02020202020202,020202
020202FE
1520 DATA 00003F202020202,202020
20203F,0000FC0404040404,04040404
04FC
1530 DATA 0000001F1010101,101010
101F,000000F808080808,08080808F8
1540 RESTORE 1500
1550 FOR BD=96 TO 111 :: READ C$
 :: CALL CHAR(BD,C$):: NEXT BD
1560 FOR BL=1 TO 4 :: CALL SPRIT
E(#BL,92+BL*4,BL+11,81,113):: NE
XT BL
1570 CALL COINC(ALL,C):: IF C TH
EN DISPLAY AT(23,1):"COINCIDENCE
" ELSE DISPLAY A
T(23,1):"PAS DE COINCIDENCE"
1580 GOTO 1570
```

Dans le deuxième exemple, bien que les coordonnées de deux lutins soient différentes, la variable "C" contient −1 puisque le second lutin chevauche légèrement le premier.

```
1590 ! *******************
1600 ! *   CALL  COINC   *
1610 ! * (ALL, variable) *
1620 ! *deuxieme  exemple*
1630 ! *******************
1640 !
1650 CALL CLEAR
1660 CALL SCREEN(4)
1670 CALL COLOR(1,16,8)
1680 CALL MAGNIFY(4)
1690 CALL CHAR(32,"1824428181422
418",96,RPT$("F",64))
1700 CALL SPRITE(#1,96,9,64,96,#
2,96,5,95,127)
1710 CALL COINC(ALL,C):: IF C TH
EN DISPLAY AT(23,1):"COINCIDENCE
" ELSE DISPLAY A
T(23,1):"PAS DE COINCIDENCE"
1720 GOTO 1710
```

Le troisième exemple met en mouvement, de manière aléatoire, 28 lutins. En suite, à chaque fois que deux ou plusieurs d'entre eux se rencontrent, le programme émet un petit "bip" sonore.

```
1730 ! *******************
1740 ! *   CALL  COINC   *
1750 ! * (ALL, variable) *
1760 ! *troisieme exemple*
1770 ! *******************
1780 !
1790 CALL CLEAR
1800 CALL SCREEN(1)
1810 CALL CHAR(40,"FFFFFFFFFFFF
FFF")
1820 DATA 3,4,5,6,7,8,9,10,11,12
,13,14,15,16,3,4,5,6,7,8,9,10,11
,12,13,14,15,16
1830 RESTORE 1820
1840 FOR B=1 TO 28 :: READ C ::
RANDOMIZE :: CALL SPRITE(#B,40,C
,100,100,INT(RND
*20)*SGN(RND-.5),INT(RND*20)*SGN
(RND-.5)):: NEXT B
1850 CALL COINC(ALL,V):: IF V TH
EN CALL SOUND(1,1000,0)
1860 GOTO 1850
```

CALL COINC (coïncidence entre deux lutins)

La syntaxe est différente de celle utilisée au paragraphe précédent :

- CALL COINC(#numéro de lutin,#numéro de lutin,tolérance,variable)

La tolérance indique au programme à partir de quelle distance entre les deux lutins il devra considérer qu'il y a coïncidence. Concrètement, si vous donnez 20 comme valeur de tolérance, et que les lutins se trouvent à moins de 20 points l'un de l'autre, la variable utilisée dans l'instruction contiendra −1, sinon, est contiendra 0. Cette fois, la détection se fait non plus de manière physique comme avec "ALL", mais de manière logique : le calcul se fait par rapport aux coordonnées des lutins. De façon générale, pour que les coïncidences soient reconnues, la valeur de tolérance doit être directement proportionnelle à la vitesse des lutins.

```
1870 ! *******************
1880 ! *   CALL  COINC   *
1890 ! *coincidence entre*
1900 ! *   deux lutins   *
1910 ! *******************
1920 !
1930 CALL CLEAR
1940 CALL MAGNIFY(2)
1950 CALL CHAR(40,"FFFFFFFFFFFF
FFF")
1960 CALL SPRITE(#1,40,9,100,100
,#2,40,5,100,100)
1970 RANDOMIZE :: CALL MOTION(#1
,INT(RND*20)*SGN(RND-.5),INT(RND
*20)*SGN(RND-.5)
,#2,INT(RND*20)*SGN(RND-.5),INT(
RND*20)*SGN(RND-.5))
1980 CALL COINC(#1,#2,20,C):: IF
 C THEN CALL SOUND(1,1000,0)
1990 GOTO 1970
```

CALL COINC (coïncidence entre un lutin et un point sur l'écran)

Cette instruction agit de la même manière que celle rencontrée dans le paragraphe précédent, le deuxième lutin est simplement remplacé par les coordonnées d'un point :

- CALL COINC (#numéro du lutin,ligne,colonne,tolérance,variable)

Le programme suivant émet un "bip" sonore chaque fois qu'il y a coïncidence théorique (en fonction de la valeur de tolérance) entre le lutin et le point visualisé par la petite croix :

```
2000 ! *******************
2010 ! *   CALL  COINC   *
2020 ! *coincidence entre*
2030 ! * un lutin et un  *
2040 ! *point sur l'ecran*
2050 ! *******************
2060 !
2070 CALL CLEAR
2080 CALL MAGNIFY(2)
2090 CALL CHAR(40,"FF81818181818
1FF101010FE10101")
2100 CALL COLOR(2,7,8)
2110 CALL HCHAR(12,16,41)
2120 CALL SPRITE(#1,40,5,100,100
)
2130 RANDOMIZE :: CALL MOTION(#1
,INT(RND*20)*SGN(RND-.5),INT(RND
*20)*SGN(RND-.5)
)
2140 CALL COINC(#1,92,124,20,C):
: IF C THEN CALL SOUND(1,1000,0)
2150 GOTO 2130
```

CALL DISTANCE (entre deux lutins)

Cette instruction, dont voici la syntaxe :

- CALL DISTANCE(#numéro de lutin,#numéro de lutin,variable)

retourne le carré de la distance entre les coins supérieurs des deux lutins. Les nombres de colonnes et de lignes séparant les lutins sont élevés au carré et additionnés. Si le produit de ce calcul est supérieur à 32767, la valeur 32767 est retournée. La distance réelle entre les lutins est la racine carrée de la valeur retournée.

```
2160 ! ******************
2170 ! *  CALL DISTANCE  *
2180 ! * distance  entre *
2190 ! *   deux lutins   *
2200 ! ******************
```

```
2210 !
2220 CALL CLEAR
2230 CALL CHAR(40,"FFFFFFFFFFFF
FFF")
2240 DISPLAY AT(24,1):"DISTANCE:
"
2250 CALL SPRITE(#1,40,9,100,100
,#2,40,5,100,100)
2260 RANDOMIZE :: CALL MOTION(#1
,INT(RND*20)*SGN(RND-.5),INT(RND
*20)*SGN(RND-.5)
,#2,INT(RND*20)*SGN(RND-.5),INT(
RND*20)*SGN(RND-.5))
2270 CALL DISTANCE(#1,#2,V):: DI
SPLAY AT(24,11):INT(SQR(V)):: GO
TO 2260
```

CALL DISTANCE (entre un lutin et un point sur l'écran)

Les conventions sont les mêmes que pour la distance entre deux lutins, sauf pour la syntaxe qui est bien sûr légèrement différente :

- CALL DISTANCE(#numéro de lutin,ligne,colonne,variable)

```
2280 ! ******************
2290 ! *  CALL DISTANCE  *
2300 ! * distance  entre *
2310 ! * un lutin et un  *
2320 ! *point sur l'ecran*
2330 ! ******************
2340 !
2350 CALL CLEAR
2360 CALL COLOR(2,7,B)
2370 CALL HCHAR(12,16,41)
2380 CALL CHAR(40,"FFFFFFFFFFFF
FFF101010FE101010")
2390 DISPLAY AT(24,1):"DISTANCE:
"
2400 CALL SPRITE(#1,40,9,100,100
)
2410 RANDOMIZE :: CALL MOTION(#1
,INT(RND*20)*SGN(RND-.5),INT(RND
*20)*SGN(RND-.5)
)
2420 CALL DISTANCE(#1,92,124,V):
: DISPLAY AT(24,11):INT(SQR(V)):
: GOTO 2410
```

CALL DELSPRITE (tous les lutins)

Cette instruction élimine tous les lutins présents sur l'écran :

- CALL DELSPRITE(ALL)

```
2430 ! ******************
2440 ! * CALL  DELSPRITE *
2450 ! *      (ALL)      *
2460 ! ******************
2470 !
2480 CALL CLEAR
2490 CALL SCREEN(1)
2500 FOR B=1 TO 28 :: RANDOMIZE
2510 CALL SPRITE(#B,B+64,INT(RND
*14)+3,INT(RND*192)+1,INT(RND*25
6)+1,INT(RND*20)
*SGN(RND-.5),INT(RND*20)*SGN(RND
-.5)):: NEXT B
2520 FOR D=0 TO 1000 :: NEXT D
2530 CALL DELSPRITE(ALL):: FOR D
=0 TO 500 :: NEXT D :: GOTO 2500
```

CALL DELSPRITE (un ou plusieurs lutins)

Cette version autorise une utilisation plus sélective que CALL DELSPRITE(ALL) :

- CALL DELPRITE(#numéro de lutin[,autre lutin])

L'exemple suivant crée 28 lutins qu'il efface ensuite un par un, en partant de celui dont le numéro est le plus élevé :

```
2540 ! ******************
2550 ! * CALL  DELSPRITE *
2560 ! *     un  lutin   *
2570 ! ******************
2580 !
2590 CALL CLEAR
2600 CALL SCREEN(1)
2610 CALL CHAR(40,"FFFFFFFFFFFF
FFF")
2620 FOR B=1 TO 28 :: RANDOMIZE
2630 CALL SPRITE(#B,40,INT(RND*1
4)+3,INT(RND*192)+1,INT(RND*256)
+1,INT(RND*20)*S
GN(RND-.5),INT(RND*20)*SGN(RND-.
5)):: NEXT B
2640 FOR D=0 TO 1000 :: NEXT D
2650 FOR B=28 TO 1 STEP -1 :: CA
LL DELSPRITE(#B):: FOR D=0 TO 20
0 :: NEXT D :: N
EXT B :: FOR D=0 TO 500 :: NEXT
D :: GOTO 2620
```

Superposition des lutins

Les lutins, ce n'est un secret pour personne, se superposent à l'écran graphique classique. En plus, si deux lutins occupent une zone d'écran commune, celui qui possède le numéro le moins élevé recouvrira l'autre :

```
2660 ! ******************
2670 ! *  SUPERPOSITION  *
2680 ! *   DES LUTINS    *
2690 ! ******************
2700 !
2710 CALL CLEAR
2720 CALL SCREEN(1)
2730 CALL MAGNIFY(2)
2740 CALL CHAR(32,"1824428181422
418FFFFFFFFFFFFFFFF")
2750 CALL COLOR(1,16,8)
2760 L=20 :: C=24
2770 DATA 2,3,5,7,4,6,9,13,10,11
,14,12,15,16,2,3,5,7,4,6,9,13,10
,11,14,12,15,16
2780 RESTORE 2770
2790 FOR B=1 TO 28 :: READ CO ::
 CALL SPRITE(#B,33,CO,L,C):: L=L
+5 :: C=C+7 :: N
EXT B
2800 GOTO 2800
```

Disparition partielle de certains lutins

Lorsque plus de 4 lutins sont situés sur la même ligne d'écran, celui ou ceux qui ont les numéros les plus élevés disparaissent :

```
2810 ! ******************
2820 ! *   DISPARITION   *
2830 ! *  PARTIELLE DE   *
2840 ! * CERTAINS LUTINS *
2850 ! ******************
2860 !
2870 CALL CLEAR
2880 CALL SCREEN(1)
2890 CALL MAGNIFY(2)
2900 CALL CHAR(32,"1824428181422
418FFFFFFFFFFFFFFFF")
2910 CALL COLOR(1,16,8)
2920 L=20 :: C=24
2930 DATA 2,3,5,7,4,6,9,13,10,11
,14,12,15,16,2,3,5,7,4,6,9,13,10
,11,14,12,15,16
2940 RESTORE 2930
2950 FOR B=1 TO 28 :: READ CO ::
 CALL SPRITE(#B,33,CO,L,C):: L=L
+5 :: C=C+7 :: N
EXT B
2960 FOR D=0 TO 1000 :: NEXT D :
: FOR B=1 TO 28 :: RANDOMIZE ::
CALL MOTION(#B,I
NT(RND*20)*SGN(RND-.5),INT(RND*2
0)*SGN(RND-.5)):: NEXT B
2970 GOTO 2970
```

Détection de la présence de cinq lutins ou plus, sur la même ligne

Aucune instruction du Basic Etendu ne permet de faire ce genre de chose. L'instruction PEEK, qui autorise la lecture du contenu d'une

adresse mémoire, nous permet de combler cette lacune.
L'adresse >837B (−31877 en décimal) contient, entre autre, le numéro du cinquième lutin s'il y a plus de quatre lutins sur la même ligne. Ce numéro est localisé dans les bits 3 à 7 (si vous désirez obtenir plus de renseignements sur cette adresse, reportez-vous à l'article de Denise Amrouche : Les utilisations de la Mini-mémoire). CALL PEEK et les opérateurs logiques AND et NOT remplacent facilement cette instruction oubliée.
Le programme suivant efface un lutin lorsque cinq lutins ou plus sont situés sur la même ligne. Le numéro du lutin effacé le plus récemment est affiché au bas de l'écran :

```
2980 ! ******************
2990 ! * DETECTION DE LA *
3000 ! *   PRESENCE DE   *
3010 ! *   CINQ LUTINS   *
3020 ! * OU PLUS, SUR LA *
3030 ! *   MEME  LIGNE   *
3040 ! ******************
3050 !
3060 CALL CLEAR
3070 CALL SCREEN(1)
3080 CALL CHAR(32,"1824428181422
418")
3090 CALL COLOR(1,5,1,3,8,1,4,8,
1)
3100 CALL MAGNIFY(2)
3110 FOR B=1 TO 28 :: RANDOMIZE
:: CALL SPRITE(#B,B+64,INT(RND*1
4)+3,INT(RND*192
)+1,INT(RND*256)+1,INT(RND*20)*S
GN(RND-.5),INT(RND*20)*SGN(RND-.
5)):: NEXT B
3120 FOR D=0 TO 1000 :: NEXT D
3130 CALL PEEK(-31877,A):: IF NO
T A AND 64 THEN 3130
3140 L=(A AND 31)+1 :: DISPLAY A
T(24,1):L :: CALL DELSPRITE(#L):
: CALL SOUND(1,1
000,0):: GOTO 3130
```

Conclusion

Nous disposons, avec les lutins, d'un système graphique puissant. Cependant, il ne faut pas se faire d'illusions; le Basic Etendu est beaucoup trop lent pour permettre la création de jeux d'action très rapides. Pour cela, il faudra passer à l'Assembleur. Les lutins sont malgré tout très pratiques pour améliorer la présentation des programmes, réaliser des simulations ou même des jeux, à condition que les déplacements soient programmés définitivement ou qu'ils soient suffisamment lents pour que les instructions de détection puissent agir. ■

Chargeur automatique en Basic Etendu

Eric Thomas

Ce petit programme, s'il est placé dans le premier lecteur de disquettes sous le nom "LOAD", permet d'exécuter directement n'importe quel programme dès que l'on accède au Basic Etendu. En effet, le programme "LOAD", s'il existe, est automatiquement exécuté. D'autre part, comme le Basic Etendu permet d'enchaîner plusieurs programmes au moyen de l'instruction "RUN", rien ne nous empêche de créer sur chaque disquette un programme affichant automatiquement son contenu (par un menu), et permettant de lancer directement l'exécution du programme choisi.

Pour que ce "chargeur" soit général et puisse fonctionner avec n'importe quelle disquette, il doit lire le fichier "DSK1." qui contient notamment les noms et types de tous les fichiers de la disquette.
Seuls sont affichés les fichiers dont l'entier déterminant le type vaut 5 ou −5, ce qui correspond à un fichier de type programme (la valeur est négative lorsque le fichier est protégé). Il ne resterait alors plus qu'à écrire "RUN A$", par exemple, où "A$" serait le nom du programme sélectionné. Mais c'est là que les choses se compliquent. Il est en effet impossible d'écrire RUN suivi d'une chaîne contenant le nom du programme à exécuter. Le TI ne peut comprendre un RUN que lorsque le nom du programme est écrit EXPLICITEMENT.

Pour garder un caractère général et fonctionner sans modification ou mise à jour d'une disquette à une autre, le programme "poke" directement le nom du programme à exécuter. Il modifie la dernière ligne du programme (RUN "DSK1.??????????) en écrivant explicitement le nom du programme choisi à la place des points d'interrogation.

Pour transformer correctement la dernière ligne, il est nécessaire de :

1) "poker" le numéro du lecteur contenant le programme (c'est la réponse à la question QUEL DISQUE ?);
2) changer l'octet indiquant au système la longueur de la chaîne située après le "RUN" (5 + la longueur du nom du programme à exécuter);
3) remplacer les points d'interrogation par les lettres composant le nom du programme;
4) "poker" un zéro derrière la dernière lettre pour marquer la fin de la ligne (les points d'interrogation résiduels sont ignorés).

ATTENTION : si vous modifiez ce programme (suppression des remarques, présentation différente du menu), il faut recalculer l'adresse du point de RUN "DSK1.??????????" (au moyen de PEEK), et l'indiquer au programme en modifiant la valeur de "M" qui est donnée ligne 200. ■

```
100 ! ********************
110 ! *     Chargeur     *
120 ! *   Basic Etendu   *
130 ! *  a usage general *
140 ! ********************
150 ! *    Copyright     *
160 ! * Gerard Santraille *
170 ! * et "99 Magazine" *
180 ! ********************
190 !
200 CALL INIT :: M=-345 :: OPTION BASE 1
:: DIM P$(20)
210 CALL SCREEN(15):: DISPLAY AT(12,8)ERA
SE ALL:"Quel disque : 1" :: ACCEPT AT(1
2,22)BEEP SIZE(-1)VALIDATE("123"):D$
220 OPEN #1:"DSK"&D$&".",INPUT ,RELATIVE,
INTERNAL :: INPUT #1:N$,A,A,A :: DISPLA
Y AT(1,1)ERASE ALL:N$;"  ";A;"secteurs li
bres"
230 M$="## ---> "&RPT$("#",10)
240 FOR I=1 TO 20
250 INPUT #1:N$,A,B,B :: IF N$="" THEN 29
0
260 IF ABS(A)=5 THEN DISPLAY AT(I+2,5):US
ING M$:I,N$ :: P$(I)=N$ ELSE 250
270 NEXT I
280 DISPLAY AT(I+2,5):USING M$:I,"CONTINU
ER"
290 DISPLAY AT(24,13):"VOTRE CHOIX :" ::
ACCEPT AT(24,27)BEEP SIZE(-2):N :: IF I
=N THEN CALL HCHAR(3,1,32,704):: GOTO 240
300 CLOSE #1 :: CALL LOAD(M-1,ASC(D$))
310 FOR P=1 TO LEN(P$(N)):: CALL LOAD(M+P
,ASC(SEG$(P$(N),P,1))):: NEXT P :: CALL
 LOAD(M-5,4+P):: CALL LOAD(M+P,0)
320 RUN "DSK1.??????????"
```

Le Solitaire

Olivier Colin

Le principe de ce jeu, écrit en **Basic TI**, est exactement celui du "solitaire". Mais il comporte des avantages : vous n'aurez pas, en jouant, à placer et à déplacer vous-même les pions, le TI étant à votre disposition pour cela. Votre seul travail consiste à entrer le numéro de la ligne et de la colonne de départ, puis à faire de même pour la case d'arrivée.

Le programme commence par l'élaboration des formes (pions, tableau et titre), aux lignes 240 à 380. Ensuite, il se branche en ligne 2360 à l'aide d'un GOSUB pour faire apparaître le titre, qui lui-même est suivi de la règle du jeu (lignes 2540 à 2640).

Les lignes 410 à 430 initialisent les variables. Puis le quadrillage se construit (lignes 480 à 640) et apparaît grâce à la routine de "transparence" (lignes 1970 à 2000), et à la routine "réapparition des couleurs" (lignes 2040 à 2070). Ce système a déjà été utilisé pour l'apparition du titre et de la règle du jeu. Les lignes 710 à 770 impriment les numéros des lignes et des colonnes (1 à 8). Le jeu commence à la ligne 810.

L'ordinateur affiche "DEPART", à l'aide d'un sous-programme expliqué plus bas, et émet un petit bip. Celui-ci vous annonce que vous pouvez entrer l'abscisse des coordonnées de départ. Dès que vous avez entré le premier chiffre à l'aide d'un CALL KEY (ce n'est donc pas la peine d'appuyer sur la touche "ENTER"), un second bip annonce que vous pouvez entrer l'ordonnée des coordonnées de départ. Après, il faut procéder de même pour l'entrée des coordonnées de la case d'arrivée du pion.

Les lignes 1300 à 1420 servent au déplacement du pion. Un caractère vide se met à la place du pion à déplacer (coordonnées de départ) qui s'affiche à la case d'arrivée. Le pion sauté est remplacé par un blanc. C'est à la ligne 1420 que le programme se branche à la ligne 2110 pour augmenter le compteur de 10 points.

Mais avant que le pion ne se déplace, il faut procéder à plusieurs opérations :

1 - vérifier que le caractère qui saute est un pion. La réponse à cette question se trouve dans les lignes 1480 à 1500 et utilise l'instruction CALL GCHAR;
2 - vérifier que le caractère d'arrivée est vide (lignes 1560 à 1640);
3 - vérifier que le caractère sauté est un pion (lignes 1700 à 1840).

Ces vérifications sont effectuées par les sous-programmes appelés au moment des entrées des coordonnées.
Le sous-programme situé aux lignes 1880 à 1930 permet d'afficher une suite de caractères (ce sous-programme correspond au "DISPLAY AT" du Basic Etendu). ■

```
100 REM ********************
110 REM *  LE  SOLITAIRE  *
120 REM ********************
130 REM *    BASIC  TI    *
140 REM ********************
150 REM *    COPYRIGHT    *
160 REM *  OLIVIER COLIN  *
170 REM * ET  99 Magazine *
180 REM ********************
190 REM********************
200 CALL CLEAR
210 REM ***************
220 REM INITIALISATION
230 REM ***************
240 CALL CHAR(97,"1010101010101010")
250 CALL CHAR(98,"00000000FF000000")
260 CALL CHAR(99,"10101010FF101010")
270 CALL CHAR(100,"000000001F101010")
280 CALL CHAR(101,"00000000FF101010")
290 CALL CHAR(102,"101010101F")
300 CALL CHAR(103,"10101010FF")
310 CALL CHAR(104,"101010101F101010")
320 CALL CHAR(105,"00000000F0101010")
330 CALL CHAR(106,"10101010F0")
340 CALL CHAR(107,"10101010F0101010")
350 CALL CHAR(108,"")
360 CALL CHAR(122,"00547C1010107CFE")
370 CALL CHAR(130,"FFFFFFFFFFFFFFFF")
380 CALL CHAR(131,"FFFCF0C0C0F0FCFF")
390 GOSUB 2360
400 CALL CLEAR
410 S=5
420 T=8
430 SCORE=0
440 GOSUB 1970
450 REM *******
460 REM TABLEAU
470 REM *******
480 PRINT TAB(20);"SCORE:";SCORE: : : :
490 PRINT TAB(8);"111dbebebi111"
500 PRINT TAB(8);"111azazaza111"
510 PRINT TAB(8);"111hbcbcbK111"
520 PRINT TAB(8);"111azazaza111"
530 PRINT TAB(7);"dbebcbcbcbcbebi"
540 PRINT TAB(7);"azazazazazazaza"
550 PRINT TAB(7);"hbcbcbcbcbcbcbK"
560 PRINT TAB(7);"azazaza azazaza"
570 PRINT TAB(7);"hbcbcbcbcbcbcbK"
580 PRINT TAB(7);"azazazazazazaza"
590 PRINT TAB(7);"fbgbcbcbcbcbgbj"
600 PRINT TAB(8);"111azazaza111"
610 PRINT TAB(8);"111hbcbcbK111"
620 PRINT TAB(8);"111azazaza111"
630 PRINT TAB(8);"111fbgbgbj111"
640 PRINT : : :
650 CALL SCREEN(11)
660 GOSUB 2040
670 CALL COLOR(12,14,1)
680 REM ******************
690 REM LIGNES ET COLONNES
700 REM ******************
710 FOR COMP=49 TO 55
720 S=S+2
730 T=T+2
740 CALL HCHAR(S,8,COMP)
750 CALL HCHAR(5,T,COMP)
760 NEXT COMP
770 PRINT TAB(17);"X=   Y=   "
780 REM *****************
790 REM ENTREZ L'ABSCISSE
800 REM *****************
810 A$="DEPART "
820 LIG=23
830 COL=3
840 GOSUB 1880
850 CALL HCHAR(23,22,30)
860 CALL HCHAR(23,27,30)
870 CALL SOUND(200,1600,0)
880 CALL KEY(0,K,S)
890 IF S<>1 THEN 880
900 IF K=1 THEN 2210
910 IF (K<49)+(K>55)THEN 880
920 CALL HCHAR(23,22,K)
930 CALL SOUND(200,1600,0)
940 CALL KEY(0,KE,ST)
950 IF ST<>1 THEN 940
960 IF KE=1 THEN 2210
970 IF (KE<49)+(KE>55)THEN 940
980 CALL HCHAR(23,27,KE)
990 A=VAL(CHR$(K))
1000 B=VAL(CHR$(KE))
1010 GOSUB 1480
1020 REM *****************
1030 REM ENTREZ L'ORDONNEE
1040 REM *****************
1050 A$="ARRIVEE"
1060 LIG=23
1070 COL=3
1080 GOSUB 1880
1090 CALL HCHAR(23,22,30)
1100 CALL HCHAR(23,27,30)
1110 CALL SOUND(200,1600,0)
1120 CALL KEY(0,K1,S1)
1130 IF S1<>1 THEN 1120
1140 IF K1=1 THEN 2210
1150 IF (K1<49)+(K1>55)THEN 1120
1160 CALL HCHAR(23,22,K1)
1170 CALL SOUND(200,1600,0)
1180 CALL KEY(0,KE1,ST1)
1190 IF ST1<>1 THEN 1180
1200 IF KE1=1 THEN 2210
1210 IF (KE1<49)+(KE1>55)THEN 1180
1220 CALL HCHAR(23,27,KE1)
1230 C=VAL(CHR$(K1))
1240 D=VAL(CHR$(KE1))
1250 GOSUB 1510
1260 GOSUB 1700
1270 REM ******************
1280 REM LE PION SE DEPLACE
1290 REM ******************
1300 CALL HCHAR(4+B*2,8+A*2,32)
1310 CALL HCHAR(4+D*2,8+C*2,122)
1320 IF A<>C THEN 1380
1330 IF KE1>KE THEN 1340 ELSE 1360
1340 CALL HCHAR(4+D*2-2,8+A*2,32)
1350 GOTO 2110
1360 CALL HCHAR(4+D*2+2,8+A*2,32)
1370 GOTO 2110
1380 IF K1>K THEN 1390 ELSE 1410
1390 CALL HCHAR(4+B*2,8+C*2-2,32)
1400 GOTO 2110
1410 CALL HCHAR(4+B*2,8+C*2+2,32)
1420 GOTO 2110
1430 REM *****************
1440 REM LE CARACTERE QUI
1450 REM SAUTE EST IL UN
1460 REM       PION ?
1470 REM *****************
1480 CALL GCHAR(4+B*2,8+A*2,CODE)
1490 IF CODE<>122 THEN 1630
1500 RETURN
1510 REM ******************
1520 REM    LE CARACTERE
1530 REM   D'ARRIVEE EST
1540 REM   IL UN VIDE??
1550 REM ******************
1560 CALL GCHAR(4+D*2,8+C*2,CODE1)
1570 IF CODE1<>32 THEN 1630
1580 IF (A<>C)*(B<>D)THEN 1630 ELSE 1590
1590 IF (A=C)*(B=D+2)THEN 1650
1600 IF (A=C)*(B=D-2)THEN 1650
1610 IF (A=C+2)*(B=D)THEN 1650
1620 IF (A=C-2)*(B=D)THEN 1650
1630 CALL SOUND(500,200,0)
1640 GOTO 810
1650 RETURN
1660 REM *****************
1670 REM LE CARACTERE SAUTE
1680 REM EST IL UN PION???
1690 REM *****************
1700 IF A<>C THEN 1780
1710 IF KE1>KE THEN 1720 ELSE 1750
```

```
1720 CALL GCHAR(4+D*2-2,8+C*2,COD)
1730 IF COD<>122 THEN 1630
1740 GOTO 1840
1750 CALL GCHAR(4+D*2+2,8+C*2,COD)
1760 IF COD<>122 THEN 1630
1770 GOTO 1840
1780 IF K1>K THEN 1790 ELSE 1820
1790 CALL GCHAR(4+D*2,8+C*2-2,COD)
1800 IF COD<>122 THEN 1630
1810 GOTO 1840
1820 CALL GCHAR(4+D*2,8+C*2+2,COD)
1830 IF COD<>122 THEN 1630
1840 RETURN
1850 REM ********
1860 REM PRINT AT
1870 REM ********
1880 REM
1890 FOR CARAC=1 TO LEN(A$)
1900 B$=SEG$(A$,CARAC,1)
1910 CALL HCHAR(LIG,COL+CARAC,ASC(B$))
1920 NEXT CARAC
1930 RETURN
1940 REM ************
1950 REM TRANSPARENCE
1960 REM ************
1970 FOR I=1 TO 16
1980 CALL COLOR(I,1,1)
1990 NEXT I
2000 RETURN
2010 REM ***************
2020 REM ROUTINE COULEUR
2030 REM ***************
2040 FOR I=1 TO 16
2050 CALL COLOR(I,2,1)
2060 NEXT I
2070 RETURN
2080 REM *****
2090 REM SCORE
2100 REM *****
2110 SCORE=SCORE+10
2120 A$=STR$(SCORE)
2130 LIG=1
2140 COL=28
2150 GOSUB 1880
2160 IF SCORE=310 THEN 2210
2170 GOTO 810
2180 REM ****************
2190 REM FIN DE LA PARTIE
2200 REM ****************
2210 CALL CLEAR
2220 PRINT "SCORE=";SCORE
2230 PRINT : : : : : : : : :
2240 PRINT "IL VOUS RESTE:";32-SCORE/10;"
 PIONS"
2250 IF SCORE<280 THEN 2280
2260 PRINT "VOUS AVEZ TRES BIEN JOUE!!!!"
2270 GOTO 2290
2280 PRINT "CE N'EST PAS UNE TRES BONNE P
ERFORMANCE!!!!": : :"MAIS VOUS FEREZ MI
EUX LA     PROCHAINE  FOIS!!!!!"
2290 PRINT "VOULEZ-VOUS REJOUER?(O/N)"
2300 CALL KEY(0,TOUCHE,ETAT)
2310 IF ETAT=0 THEN 2300
2320 IF (TOUCHE<>79)*(TOUCHE<>78)THEN 230
0
2330 IF TOUCHE=79 THEN 400
2340 CALL CLEAR
2350 END
2360 GOSUB 1970
2370 A$="JEU  DU"
2380 LIG=6
2390 COL=12
2400 GOSUB 1880
2410 FOR LIRE=1 TO 58
2420 READ A1,A2,A3,A4
2430 CALL HCHAR(A1,A2,A3,A4)
2440 NEXT LIRE
2450 A$="UNE TOUCHE POUR CONTINUER"
2460 LIG=24
2470 COL=3
2480 GOSUB 1880
2490 GOSUB 2040
2500 CALL KEY(0,KEY,STATUS)
2510 IF STATUS=0 THEN 2500
2520 CALL CLEAR
2530 GOSUB 1970
2540 PRINT "LE JEU CONSISTE A FAIRE      D
ISPARAITRE TOUTES LES       FICHES.":"LE
S FICHES SE DEPLACENT  "
2550 PRINT "HORIZONTALEMENT OU VERTI-   C
ALEMENT ET PRENNENT LE      PION A CONDI
TION QU'IL Y AIT"
2560 PRINT "UN TROU POUR SE DEPLACER,   C
OMME LE JEU DE DAMES.": :"CELLE QUI VIE
NT D'ETRE      "
2570 PRINT "SAUTEE EST ENLEVEE. LE JEU  E
ST·TERMINE LOSQU'IL NE      RESTE QU'UNE
 FICHE": :
2580 PRINT "NEANMOINS, SI VOUS N'AVEZ   P
LUS DE POSSIBILITE DE       SAUTER, ARRE
TEZ LE JEU EN   "
2590 PRINT "FAISANT FCTN 7(AID)."
2600 PRINT : :"UNE TOUCHE POUR CONTINUER"
2610 GOSUB 2040
2620 CALL KEY(0,TOU,ETA)
2630 IF ETA=0 THEN 2620
2640 RETURN
2650 END
2660 DATA 1,1,130,3,2,1,130,1,3,1,130,3,4
,3,130,1,5,1,130,3,5,5,130,3,6,5,130,1,
7,5,130,1
2670 DATA 8,5,130,1,9,5,130,3,6,7,130,1,7
,7,130,1,8,7,130,1,9,9,130,1,10,9,130,1
,11,9,130,1
2680 DATA 12,9,130,1,13,9,130,3,13,13,130
,1,14,13,130,1,15,13,130,1,16,13,130,1,
17,13,130,1,11,13,130,1
2690 DATA 17,15,130,3,18,16,130,1,19,16,1
30,1,19,16,130,1,20,16,130,1,21,16,130,
1,13,19,130,3
2700 DATA 14,19,130,1,14,21,130,1,15,19,1
30,3,16,19,130,1,16,21,130,1,17,19,130,
1,17,21,130,1
2710 DATA 9,23,130,1,10,23,130,1,11,23,13
0,1,12,23,130,1,13,23,130,1,7,23,130,1
2720 DATA 5,25,130,3,6,25,130,1,6,27,130,
1,7,25,130,2,7,27,131,1,8,25,130,1,8,27
,130,1,9,25,130,1
2730 DATA 9,27,130,1,1,29,130,3,2,29,130,
1,3,29,130,2,4,29,130,1,5,29,130,3
```

Claustrophobia

Julien Thomas

Nos amis belges du club TISOFT-HOME nous envoient encore un très bon programme, écrit en **Basic TI**.

Le but de ce jeu est d'arriver à encercler un horrible monstre, à l'aide de briques disposées au hasard sur l'écran. Vous êtes représenté par une petite "bestiole" ni plus ni moins horrible que la chose à encercler. Les déplacements se font avec les touches fléchées, ce qui permet de déplacer les briques. Quand le monstre ne pourra plus bouger, vous devrez appuyer rapidement sur la touche "C". Le jeu semble simple mais l'auteur a ajouté quelques astuces qui compliquent les choses :

- les briques vertes sont fixes;
- de temps en temps, le monstre disparaît et réapparaît à un tout autre endroit de façon imprévisible. Le choix du niveau de jeu conditionne la fréquence des disparitions;
- si vous vous en approchez trop, le monstre peut vous dévorer.

Signalons enfin que le programme permet de fixer le nombre de briques à disposer sur l'écran. En choisissant un niveau de jeu élevé, il devient très difficile de "coincer la petite bête". Amusez-vous bien...

```
100 REM ******************
110 REM * CLAUSTROPHOBIA  *
120 REM ******************
130 REM *    BASIC TI     *
140 REM ******************
150 REM *    COPYRIGHT    *
160 REM *   TISOFT-HOME   *
170 REM * ET  99 Magazine *
180 REM ******************
190 CALL CLEAR
200 CALL SCREEN(5)
210 CALL CHAR(128,"007E7E7E7E7E7E00")
220 CALL COLOR(13,10,5)
230 CALL SOUND(500,262,0,523,0)
240 CALL HCHAR(3,3,128,3)
250 CALL VCHAR(4,3,128,4)
260 CALL HCHAR(7,4,128,2)
270 CALL SOUND(500,294,0,587,0)
280 CALL VCHAR(9,6,128,5)
290 CALL HCHAR(13,7,128,2)
300 CALL SOUND(500,330,0,659,0)
310 CALL HCHAR(1,9,128,3)
320 CALL VCHAR(2,9,128,4)
330 CALL VCHAR(2,11,128,4)
340 CALL HCHAR(3,10,128)
350 CALL SOUND(500,349,0,698,0)
360 CALL VCHAR(2,15,128,5)
370 CALL VCHAR(2,17,128,5)
380 CALL HCHAR(6,16,128)
390 CALL SOUND(500,392,0,784,0)
400 CALL HCHAR(8,13,128,3)
410 CALL VCHAR(9,13,128,2)
420 CALL HCHAR(10,14,128,2)
430 CALL VCHAR(11,15,128,2)
440 CALL HCHAR(12,13,128,2)
450 CALL SOUND(500,440,0,880,0)
460 CALL HCHAR(6,21,128,3)
470 CALL VCHAR(7,22,128,4)
480 CALL SOUND(500,494,0,988,0)
490 CALL HCHAR(2,26,128,3)
500 CALL VCHAR(3,26,128,4)
510 CALL VCHAR(3,28,128,2)
520 CALL VCHAR(4,27,128,2)
530 CALL HCHAR(6,28,128)
540 CALL SOUND(500,523,0,131,0)
550 CALL HCHAR(10,27,128,3)
560 CALL VCHAR(11,27,128,4)
570 CALL VCHAR(11,29,128,4)
580 CALL HCHAR(14,28,128)
590 CALL SOUND(500,587,0,147,0)
600 CALL HCHAR(17,3,128,3)
610 CALL VCHAR(18,3,128,4)
620 CALL HCHAR(19,4,128)
630 CALL SOUND(500,659,0,165,0)
640 CALL HCHAR(15,9,128,3)
650 CALL VCHAR(16,9,128,4)
660 CALL VCHAR(16,11,128,4)
```

```
670 CALL HCHAR(19,10,128)
680 CALL SOUND(500,698,0,175,0)
690 CALL HCHAR(16,16,128,3)
700 CALL VCHAR(17,16,128,4)
710 CALL HCHAR(17,18,128)
720 CALL HCHAR(18,17,128,3)
730 CALL HCHAR(19,19,128)
740 CALL HCHAR(20,17,128,3)
750 CALL SOUND(500,784,0,196,0)
760 CALL VCHAR(14,22,128,5)
770 CALL SOUND(500,880,0,220,0)
780 CALL HCHAR(17,26,128,3)
790 CALL VCHAR(18,26,128,4)
800 CALL VCHAR(18,28,128,4)
810 CALL HCHAR(19,27,128)
820 CALL SOUND(500,988,0,247,0)
830 CALL SOUND(500,1047,0,262,0)
840 FOR A=1 TO 1000
850 NEXT A
860 CALL CLEAR
870 PRINT TAB(8);"CLAUSTROPHOBIA"
880 PRINT TAB(7);"================"
890 PRINT :"ESSAYEZ DE CAPTURER L'ENNEMIE
N L'ENCERCLANT AVEC LES    BRIQUES."
900 PRINT "VOUS POUVEZ POUSSER HORIZON-TA
LEMENT OU VERTICALEMENT"
910 PRINT "UNE OU PLUSIEURS BRIQUES.    MA
IS PRENEZ GARDE !!!"
920 PRINT "LES BRIQUES VERTES NE        PE
UVENT PAS ETRE DEPLACEES"
930 PRINT "ET L'ENNEMI PEUT BRUSQUEMENTCH
ANGER DE POSITION."
940 PRINT "UTILISEZ LES FLECHES POUR    VO
US DEPLACER ET APPUYEZ SUR"
950 PRINT "'C' QUAND VOUS AUREZ         CA
PTURE L'ENNEMI. (FAITES LERAPIDEMENT)."
960 PRINT : :"POUR CONTINUER, APPUYEZ SUR
 N'IMPORTE QUELLE TOUCHE..."
970 CALL KEY(0,K,S)
980 IF S=0 THEN 970
990 CALL CLEAR
1000 PRINT "LE JEU OFFRE LA POSSIBILITE D
E CHOISIR ENTRE 10 NIVEAUX."
1010 PRINT :"DEUX POINTS VARIENT SELON LE
NIVEAU:":
1020 PRINT :"1) NOMBRE DE BRIQUES VERTES.
2) FREQUENCE DES CHANGEMENTS   DE POSIT
ION DE L'ENNEMI."
1030 PRINT :"IL EXISTE AUSSI UN NIVEAU
POUR LES ENFANTS (NIVEAU 0)."
1040 PRINT :"DANS CE CAS, LES POINTS 1 ET
2 NE S'APPLIQUENT PAS."
1050 PRINT :"BONNE CHANCE !!!"
1060 PRINT : : :"POUR COMMENCER, APPUYEZ
SUR N'IMPORTE QUELLE TOUCHE..."
1070 CALL KEY(0,K,S)
1080 IF S=0 THEN 1070
1090 CALL CLEAR
1100 PRINT "NIVEAU DE DIFFICULTE"
1110 PRINT
1120 INPUT "(0-10)              ":DIF
1130 PRINT
1140 PRINT
1150 IF (DIF>10)+(DIF<0)THEN 1100
1160 PRINT "NOMBRE DE BRIQUES:"
1170 PRINT
1180 INPUT "(MAX.400)           ":QUA
1190 IF (QUA>400)+(QUA<4)THEN 1130
1200 CALL CLEAR
1210 HYPER=0
1220 TIME=0
1230 IF DIF=0 THEN 1260
1240 TEST=(11-DIF)*10
1250 GOTO 1270
1260 TEST=1000
1270 CALL CHAR(136,"FFFFFFFFFFFFFFFF")
1280 CALL CHAR(137,"007E7E7E7E7E7E00")
1290 CALL CHAR(144,"99997E3C3C7E9981")
1300 CALL CHAR(145,"C3243C7F7F3C24C3")
1310 CALL CHAR(146,"81997E3C3C7E9999")
1320 CALL CHAR(147,"C3243CFEFE3C24C3")
1330 CALL CHAR(152,"3C7EFBFFF8F0793E")
1340 CALL CHAR(96,"18187E7E18181818")
1350 CALL COLOR(14,3,5)
1360 CALL COLOR(15,2,5)
1370 CALL COLOR(16,12,5)
1380 CALL CLEAR
1390 CALL HCHAR(1,2,136,30)
1400 CALL VCHAR(2,2,136,22)
1410 CALL VCHAR(2,31,136,22)
1420 CALL HCHAR(24,2,136,30)
1430 RANDOMIZE
1440 IF DIF=0 THEN 1520
1450 FOR A=1 TO 2*DIF
1460 X=INT(RND*22)+2
1470 Y=INT(RND*24)+5
1480 CALL GCHAR(X,Y,C)
1490 IF C=137 THEN 1460
1500 CALL HCHAR(X,Y,137)
1510 NEXT A
1520 FOR A=1 TO QUA
1530 X=INT(RND*22)+2
1540 Y=INT(RND*24)+5
1550 CALL GCHAR(X,Y,C)
1560 IF C<>32 THEN 1530
1570 CALL HCHAR(X,Y,128)
1580 NEXT A
1590 RICH=4
1600 MX=INT(RND*22)+2
1610 MY=30
1620 CALL HCHAR(MX,MY,152)
1630 X=INT(RND*22)+2
1640 Y=3
1650 CALL HCHAR(X,Y,145)
1660 FOR A=1 TO 3
1670 CALL SOUND(100,523,0)
1680 CALL SOUND(100,659,0)
1690 NEXT A
1700 CALL SOUND(400,523,0)
1710 CALL KEY(0,KEY,STATUS)
1720 TIME=TIME+1
1730 IF STATUS=0 THEN 2660
1740 IF KEY=69 THEN 1790
1750 IF KEY=68 THEN 2040
1760 IF KEY=88 THEN 1810
1770 IF KEY=83 THEN 2020
1780 GOTO 2660
1790 DIR=-1
1800 GOTO 1820
1810 DIR=1
1820 CALL SOUND(30,500,0)
1830 TEL=0
1840 CALL GCHAR(X+DIR,Y,A)
1850 IF A=32 THEN 1900
1860 IF (A=136)+(A=137)+(A=152)THEN 1980
1870 X=X+DIR
1880 TEL=TEL+DIR
1890 GOTO 1840
1900 IF TEL=0 THEN 1930
1910 IF A=136 THEN 1980
1920 CALL HCHAR(X+DIR,Y,128)
1930 X=X-TEL+DIR
1940 CALL SOUND(50,-1,0)
1950 CALL HCHAR(X-DIR,Y,32)
1960 CALL HCHAR(X,Y,145+DIR)
1970 GOTO 2660
1980 X=X-TEL
1990 IF A=152 THEN 2250
2000 CALL SOUND(100,110,0)
2010 GOTO 2660
2020 DIR=-1
2030 GOTO 2050
2040 DIR=1
2050 CALL SOUND(30,500,0)
2060 TEL=0
2070 CALL GCHAR(X,Y+DIR,A)
2080 IF A=32 THEN 2130
2090 IF (A=136)+(A=137)+(A=152)THEN 2210
2100 Y=Y+DIR
2110 TEL=TEL+DIR
2120 GOTO 2070
2130 IF TEL=0 THEN 2160
2140 IF A=136 THEN 1980
2150 CALL HCHAR(X,Y+DIR,128)
2160 Y=Y-TEL+DIR
2170 CALL SOUND(50,-1,0)
2180 CALL HCHAR(X,Y-DIR,32)
2190 CALL HCHAR(X,Y,146-DIR)
2200 GOTO 2660
2210 Y=Y-TEL
2220 IF A=152 THEN 2340
2230 CALL SOUND(100,110,0)
2240 GOTO 2660
2250 X=X+DIR
2260 IF TEL=0 THEN 2320
2270 CALL SOUND(10,500,0)
2280 CALL SOUND(10,500,0)
2290 CALL HCHAR(X-DIR,Y,32)
2300 CALL HCHAR(X,Y,145+DIR)
2310 GOTO 2660
2320 CALL HCHAR(X-DIR,Y,32)
2330 GOTO 2430
2340 Y=Y+DIR
2350 IF TEL=0 THEN 2410
2360 CALL SOUND(10,500,0)
2370 CALL SOUND(10,500,0)
2380 CALL HCHAR(X,Y-DIR,32)
2390 CALL HCHAR(X,Y,146-DIR)
2400 GOTO 2660
2410 CALL HCHAR(X,Y-DIR,32)
2420 GOTO 2430
2430 CALL HCHAR(X,Y,96)
2440 DATA 500,262,375,262,125,262,500,262
,375,311,125,294,250,294,250,262,250,26
2,250,233,500,262
2450 RESTORE 2440
2460 FOR A=1 TO 11
2470 READ T
2480 READ F
2490 CALL SOUND(T,F,0)
2500 NEXT A
2510 CALL CLEAR
2520 PRINT TAB(10);"VOUS PERDEZ"
2530 FOR A=1 TO 12
2540 PRINT
2550 NEXT A
2560 PRINT TAB(4);"VOULEZ VOUS REJOUER ?
(O/N)"
2570 CALL KEY(0,K,S)
2580 IF (K<>78)*(K<>79)THEN 2570
2590 IF K=78 THEN 3360
2600 GOTO 1090
2610 CALL KEY(0,K,S)
2620 IF K=67 THEN 2930
2630 HYPER=HYPER+1
2640 IF HYPER>TEST THEN 3210
2650 RICH=INT(RND*4)+1
2660 ON RICH GOTO 2670,2800,2670,2800
2670 ZIN=INT(RICH-1.5)
2680 CALL GCHAR(MX+ZIN,MY,CH)
2690 IF CH<>32 THEN 2750
2700 CALL HCHAR(MX,MY,32)
2710 CALL SOUND(30,-7,0)
2720 MX=MX+ZIN
2730 CALL HCHAR(MX,MY,152)
2740 GOTO 1710
2750 IF (CH<144)+(CH>147)THEN 2610
2760 CALL HCHAR(MX,MY,32)
2770 CALL HCHAR(MX+ZIN,MY,152)
2780 CALL SOUND(1000,-7,0)
2790 GOTO 2450
2800 ZIN=-(INT(RICH-2.5))
2810 CALL GCHAR(MX,MY+ZIN,CH)
2820 IF CH<>32 THEN 2880
2830 CALL HCHAR(MX,MY,32)
2840 CALL SOUND(30,-7,0)
2850 MY=MY+ZIN
2860 CALL HCHAR(MX,MY,152)
2870 GOTO 1710
2880 IF (CH<144)+(CH>147)THEN 2610
2890 CALL HCHAR(MX,MY,32)
2900 CALL HCHAR(MX,MY+ZIN,152)
2910 CALL SOUND(1000,-7,0)
2920 GOTO 2450
2930 CALL GCHAR(MX-1,MY,N)
2940 CALL GCHAR(MX,MY+1,E)
2950 CALL GCHAR(MX+1,MY,S)
2960 CALL GCHAR(MX,MY-1,W)
2970 IF (N<>136)*(N<>128)THEN 3020
2980 IF (E<>136)*(E<>128)THEN 3020
2990 IF (S<>136)*(S<>128)THEN 3020
3000 IF (W<>136)*(W<>128)THEN 3020
3010 GOTO 3070
3020 N=0
3030 E=0
3040 S=0
3050 W=0
3060 GOTO 1710
3070 CALL HCHAR(MX,MY,96)
3080 FOR F=1000 TO 110 STEP -40
3090 CALL SOUND(-200,F,0)
3100 NEXT F
3110 FOR A=1 TO 1000
3120 NEXT A
3130 CALL CLEAR
3140 PRINT TAB(10);"VOUS GAGNEZ"
3150 PRINT
3160 PRINT TAB(10);"TEMPS =";TIME
3170 FOR A=1 TO 10
3180 PRINT
3190 NEXT A
3200 GOTO 2560
3210 CALL HCHAR(MX,MY,32)
3220 MX=INT(RND*22)+2
3230 MY=INT(RND*30)+2
3240 CALL GCHAR(MX-1,MY,N)
3250 CALL GCHAR(MX,MY-1,W)
3260 CALL GCHAR(MX,MY,CH)
3270 CALL GCHAR(MX,MY+1,E)
3280 CALL GCHAR(MX+1,MY,S)
3290 IF (CH<>32)+((N<>32)*(E<>32)*(S<>32)
*(W<>32))THEN 3220
3300 CALL HCHAR(MX,MY,152)
3310 FOR F=110 TO 1000 STEP 70
3320 CALL SOUND(-100,F,0)
3330 NEXT F
3340 HYPER=0
3350 GOTO 1710
3360 CALL CLEAR
3370 END
```

"DISPLAY" et "ACCEPT" avec le module "PRK"

Sophie Ehster

Une importante découverte sur le module "Personal record keeping" (PRK), que l'on trouve en version française sous le nom de "gestion de fichiers".
Heureux les possesseurs de "PRK", puisqu'ils disposent de deux nouvelles instructions accessibles à partir du TI Basic. En effet, lorsque le module est connecté à la console, on peut se servir de deux nouveaux sous-programmes dont les rôles sont sensiblement les mêmes que ceux des instructions "ACCEPT AT" et "DISPLAY AT" du Basic Etendu.

CALL A

Le premier s'appelle tout simplement A. Il permet de saisir une donnée (comme le fait INPUT), mais en un point quelconque de l'écran et avec un contrôle optionnel des bornes de la variable. La syntaxe de cette instruction est la suivante :

CALL A(Nligne,Ncolonne,Longmax,FCTNval,Variable,Min,Max)

chaque argument ayant la signification suivante :

- Nligne : numéro de la ligne (1 à 24)
- Ncolonne : numéro de la colonne (1 à 32)
- Longmax : longueur maximale de la variable saisie
- FCTNval : code de retour si la touche FCTN associée à certaines touches est pressée
- Variable : variable saisie
- Min : borne inférieure
- Max : borne supérieure

Codes de fonction retournés par A

- CLEAR (FCTN 4) : 2
- BEGIN (FCTN 5) : 6
- PROC'D (FCTN 6) : 5
- AID (FCTN 7) : 3
- REDO (FCTN 8) : 4
- BACK (FCTN 9) : 7
- ENTER : 1

Remarques :

1) Ces codes n'ont rien à voir avec ceux retournés par KEY qui correspondent, eux, aux valeurs ASCII des fonctions;

2) "CLEAR" sert à "nettoyer" la saisie en cours, le champ (dont la longueur est fixée par Longmax) est effacé, et le curseur est repositionné au début de celui-ci. Lorsque le champ est vide, et qu'on le valide par "ENTER", FCTNval vaut 2 (cf tableau). "CLEAR" ne permet pas, comme les autres fonctions, de sortir directement de la saisie. Il doit être validé par "ENTER";
Attention : "CLEAR" ne peut donc plus interrompre le programme lors de l'attente d'une saisie;

3) Une pression sur "ENTER" retourne (dans FCTNval) une valeur de 1 (à moins que le champ ne soit vide, comme on vient de le voir). Ainsi, lorsqu'on rentre une variable (numérique ou chaîne) que l'on valide par "ENTER", FCTNval vaut également 1;

4) Une variable numérique en cours de saisie n'est pas modifiée si l'on presse "ENTER" lorsque le champ est vide (action rejetée par "INPUT", qui écrit "*WARNING INPUT ERROR IN XXX").
Ainsi :

```
10 CALL CLEAR
20 A=66
30 CALL A(10,10,3,F,A)
40 PRINT A
```

écrira 66 (valeur inchangée), si on presse simplement ENTER;

5) Le contrôle du domaine de variation n'étant pas obligatoire, on peut omettre "Min" et "Max". S'ils sont présents, et que l'on tente d'entrer une variable hors de ces limites, un son grave est émis lorsque l'on presse "ENTER". La valeur n'est pas prise en compte, et le curseur se repositionne à son emplacement initial. Il est bien entendu impossible de spécifier des bornes lorsque A est utilisée pour la saisie de chaînes;

6) Lorsque la variable saisie est numérique, un son grave est émis si l'on presse une touche autre que 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, –, +, . ou E.

Exemple d'utilisation

CALL A(16,10,2,F,VAR,17,34)

permet de saisir la variable VAR sur 2 chiffres, à la seizième ligne et à la dixième colonne, avec un contrôle pour vérifier qu'elle est bien comprise entre 17 et 34. Si on presse "FCTN" (associée à une autre touche) au lieu de rentrer VAR, le code de la fonction sera stocké dans F.

CALL D

Le deuxième sous-programme se nomme D, et permet d'afficher une chaîne ou une variable, en un point quelconque de l'écran. Cette nouvelle instruction, tout comme la précédente, est très intéressante puisque le Basic TI ne possède pas le classique "PRINT AT", pourtant assez répandu par ailleurs.
La syntaxe est la suivante :

CALL D(Nligne,Ncolonne,Longmax,Variable)

chacun de ces arguments ayant la même signification que pour A.

L'utilisation de ces deux nouveaux sous-programmes permet donc d'entrer une variable (numérique ou chaîne) en une position quelconque de l'écran, avec un message d'invitation, un contrôle de la saisie et enfin la possibilité de prendre une valeur par défaut.

```
10 CALL D(1,5,15,"Valeur de X :24")
20 CALL A(1,18,2,F,X,10,30)
```

positionne le curseur sur le 2 de 24, et attend l'introduction de X sur deux chiffres. Si l'on appuie sur "ENTER" sans rien écrire, X vaudra 24. Si l'on appuie sur "FCTN" et 7, alors F vaudra 3.

Ces deux instructions additionnelles permettent d'augmenter notablement l'ergonomie des programmes. Pour une exploitation de tableaux de données du type de celle de "PRK", ils sont indispensables.
De nombreuses ressources se cachent vraisemblablement au fond de plusieurs autres modules. A vous de les traquer, et n'oubliez pas de nous faire part de vos découvertes : elles intéressent tous les utilisateurs de TI-99.

■

Une erreur dans le manuel de la Mini-mémoire

Jean-Luc Bazanegue

A la page 72, consacrée à l'organisation de la mémoire morte de la Mini-mémoire, il est dit que la table des références prédéfinies est située à l'adresse >6F38. En fait, elle débute un peu plus bas, à l'adresse >6F0E. Rappelons au passage que chaque référence occupe huit octets, six pour les codes ASCII constituant le symbole (si le symbole utilise moins de six caractères, comme "ERR", les octets restant sont remplis par le caractère "espace") et deux octets pour l'adresse réelle. Vous trouverez ci-dessous le contenu de la table.

```
28430 >6F0E >55 >54 UTLTAB
28432 >6F10 >4C >54
28434 >6F12 >41 >42
28436 >6F14 >7020
28438 >6F16 >50 >41 PAD
28440 >6F18 >44 >20
28442 >6F1A >20 >20
28444 >6F1C >8300
28446 >6F1E >47 >50 GPLWS
28448 >6F20 >4C >57
28450 >6F22 >53 >20
28452 >6F24 >83E0
28454 >6F26 >53 >4F SOUND
28456 >6F28 >55 >4E
28458 >6F2A >44 >20
28460 >6F2C >8400
28462 >6F2E >56 >44 VDPRD
28464 >6F30 >50 >52
28466 >6F32 >44 >20
28468 >6F34 >8800
28470 >6F36 >56 >44 VDPSTA
28472 >6F38 >50 >53
28474 >6F3A >54 >41
28476 >6F3C >8802
28478 >6F3E >56 >44 VDPWD
28480 >6F40 >50 >57
28482 >6F42 >44 >20
28484 >6F44 >8C00
28486 >6F46 >56 >44 VDPWA
28488 >6F48 >50 >57
28490 >6F4A >41 >20
28492 >6F4C >8C02
28494 >6F4E >53 >50 SPCHRD
28496 >6F50 >43 >48
28498 >6F52 >52 >54
28500 >6F54 >9000
28502 >6F56 >53 >50 SPCHWT
28504 >6F58 >43 >48
28506 >6F5A >57 >54
28508 >6F5C >9400
28510 >6F5E >47 >52 GRMRD
28512 >6F60 >4D >52
28514 >6F62 >44 >20
28516 >6F64 >9800
28518 >6F66 >47 >52 GRMRA
28520 >6F68 >4D >52
28522 >6F6A >41 >20
28524 >6F6C >9B02
28526 >6F6E >47 >52 GRMWD
28528 >6F70 >4D >57
28530 >6F72 >44 >20
28532 >6F74 >9C00
28534 >6F76 >47 >52 GRMWA
28536 >6F78 >4D >57
28538 >6F7A >41 >20
28540 >6F7C >9C02
28542 >6F7E >53 >43 SCAN
28544 >6F80 >41 >4E
28546 >6F82 >20 >20
28548 >6F84 >000E
28550 >6F86 >58 >4D XMLLNK
28552 >6F88 >4C >4C
28554 >6F8A >4E >4B
28556 >6F8C >601C
28558 >6F8E >4B >53 KSCAN
28560 >6F90 >43 >41
28562 >6F92 >4E >20
28564 >6F94 >6020
28566 >6F96 >56 >53 VSBW
28568 >6F98 >42 >57
28570 >6F9A >20 >20
28572 >6F9C >6024
28574 >6F9E >56 >4D VMBW
28576 >6FA0 >42 >57
28578 >6FA2 >20 >20
28580 >6FA4 >6028
28582 >6FA6 >56 >53 VSBR
28584 >6FA8 >42 >52
28586 >6FAA >20 >20
28588 >6FAC >602C
28590 >6FAE >56 >4D VMBR
28592 >6FB0 >42 >52
28594 >6FB2 >20 >20
28596 >6FB4 >6030
28598 >6FB6 >56 >57 VWTR
28600 >6FB8 >54 >52
28602 >6FBA >20 >20
28604 >6FBC >6034
20606 >6FBE >44 >53 DSRLNK
28608 >6FC0 >52 >4C
28610 >6FC2 >4E >4B
28612 >6FC4 >6038
28614 >6FC6 >4C >4F LOADER
28616 >6FC8 >41 >44
28618 >6FCA >45 >52
28620 >6FCC >603C
28622 >6FCE >47 >50 GPLLNK
28624 >6FD0 >4C >4C
28626 >6FD2 >4E >4B
28628 >6FD4 >6018
28630 >6FD6 >4E >55 NUMASG
28532 >6FD8 >4D >41
28634 >6FDA >53 >47
28636 >6FDC >6040
28638 >6FDE >4E >55 NUMREF
28640 >6FE0 >4D >52
28642 >6FE2 >45 >46
28644 >6FE4 >6044
28646 >6FE6 >53 >54 STRASG
28648 >6FE8 >52 >41
28650 >6FEA >53 >47
28652 >6FEC >6048
28654 >6FEE >53 >54 STRREF
28656 >6FF0 >52 >52
28658 >6FF2 >45 >46
28660 >6FF4 >604C
28662 >6FF6 >45 >52 ERR
28664 >6FF8 >52 >20
28666 >6FFA >20 >20
28668 >6FFC >6050
```

Petites annonces (gratuites)

Vends (toutes propositions acceptées) ordinateur TI-99/4A, câble pour deux magnétophones, magnétocassette Thomson, manettes de jeu. Modules Munch Man, Parsec, The Attack, TI-Invaders, Gestion de fichier, Yahtzee, Teach yourself beginning Basic. Cassettes Oldies but Goldies et Teach yourself Extended Basic. Livres "à la découverte du TI-99", "Pratique du TI-99" et "Initiation au Basic". Docteur Blaisot-Le-Stanguennec - Tél : 473 00 51.

Vends Boîtier d'extension, carte 32K, carte série/par., contrôleur de disquettes, lecteur de disquettes, imprimante GP100 câble. Contacter le journal qui transmettra.

Cherche Modules Basic Etendu, TI-Invadeurs, Jeu rétro I et II. Gérard Lebon - Tél : (8) 322 84 23.

Achète module Basic Etendu. Monsieur Albert Saradin - 3, rue Michel Venini - Quincey - 70000 Vesoul.

Recherche lecteur de disquette pour TI-99. Possède contrôleur + boîtier d'extension + TI-Writer. Marcel Stoecklin - 2, Square des Acacias - 60400 Noyon - Tél : (4) 444 08 46.

Echangerais programmes et astuces.

Cherche livre d'occasion (mais en bon état) sur le TI-99/4A. Cyril Lerat - 11, Rue de Lorraine - Vert Galand - 93410 Vaujours - Tél : 860 73 41.

Cherche module Mini-mémoire pour assembleur - Stanislas Blanchy -4, Rue Léo Delibes - 75116 Paris - Tél : (1) 704 43 17.

Achète d'urgence une carte contrôleur de disquette pour TI-99. Monsieur Marc Bayet - 1, Rue Neyret - 69283 Lyon Cedex 1.

Vends ou échange Editeur/Assembleur (jamais utilisé) et échange module Tombstone City.

Téléphoner à partir de 17 heures (sauf le lundi), Fernando Pico au 666 73 54.

Recherche des programmes de mathématiques, physique, chimie et gestion. Possibilité d'échange. Retour des documents si demandé. Je suis lycéen en 1ère S. Je cherche aussi des possesseurs de TI ou club informatique dans la région Charente Maritime. Philippe Boursier - 17, Rue de la Pierre Folle - 17270 Montguyon.

Recherche module Basic Etendu. David Cardon - Tél : (20) 29 06 21.

Echange ou vends des programmes et 3 modules de jeux (Hustle, Wumpus et Football) contre Basic Etendu, Mini-mémoire ou autres propositions. Rudy Hoevenaeghel - Rue Courte, 14 - 7780 Comines - Belgique.

Achète Basic Etendu, Blasto, Car Wars, Football, Scrabble. Alain - Tél : 525 34 18. ■

Réparez votre Assembleur ligne par ligne

Gilles Pavy

Si vous utilisez la Mini-mémoire, vous vous êtes sans doute rendus compte que les programmes NEW et OLD de l'assembleur ligne par ligne ne présentaient pas les différences annoncées dans le mode d'emploi. Afin d'y remédier, faites les modifications indiquées dans le listing. EASY BUG permet de réaliser ces modifications très simplement.

Sauvegardez votre nouvel assembleur sur cassette. Lorsque vous devrez interrompre la saisie d'un programme, vous pourrez couper l'alimentation du TI. Si, au moment où vous reprendrez votre travail, vous répondez OLD à la question PROGRAM NAME, vous retrouverez votre table des étiquettes ainsi que votre première adresse de travail. ■

```
Adresse  Code  Etiquette  Instruction  Operande   Commentaire
-------  ----  ---------  -----------  --------   -----------

>71A6    04E0  NEW        CLR          @>713E     a remplacer par CLR @>7CD6
>71A8    713E
>71AA    1002             JMP          >71B0
>71AC    0720  OLD        SETO         @>713E     a remplacer par SETO @>7CD6
>71AE    713E
>71B0    C80B             MOVE         R11,@>7182
>71B2    7182
  .
  .
  .
>7226    0560             INV          @>713E     a remplacer par INV @>7CD6
>7228    713E
>722A    1307             JEQ          >723A
>722C    0200             LI           R0,>7D00   a modifier si vous voulez chan-
>722E    7D00                                     ger l'adresse pointee par NEW
>7230    04E0             CLR          @>7190
>7232    7190
>7234    04E0             CLR          @>718E
>7236    718E
>7238    C240             MOV          R0,R9
>723A    0420             BLWP         @>79D2
>723C    79D2
>723E    7972             DATA         >7972
  .
  .
  .
```

Courrier des lecteurs

Alexandre Duback

Je vous écris afin que vous m'aidiez à résoudre le petit problème que voici : tous les programmes sont décalés vers la partie gauche du téléviseur, et une partie n'apparaît pas. De quoi cela provient-il, et que faire pour y remédier ?
A propos de "99 Magazine" : je le trouve très bon.

Eric Dauris - 13 Rue Hélène Boucher - 17300 Rochefort

Tous TI-99 provoquent ce décalage fort gênant, surtout si l'on programme en Assembleur ou en Pascal car, dans ce cas, la première colonne de caractères n'est pas visible. Il n'est pas possible (du moins pas simplement) d'agir sur l'ordinateur pour recadrer l'image. Par contre, les réglages du téléviseur peuvent être modifiés, afin d'obtenir un cadrage satisfaisant. N'importe quel technicien peut effectuer le réglage de votre récepteur en quelques minutes. Si vous faites réaliser ces réglages, les images télévisées seront décalées vers la droite, mais ceci n'est visible que dans des cas particuliers (mires, génériques ou images très géométriques).

Tout d'abord, je dois vous dire que je trouve "99 Magazine" très bien structuré et que les programmes y sont de très bonne qualité. Mais je vous écris surtout pour vous donner un autre tuyau au sujet des manettes de jeu (numéro 3 de "99").
J'ai eu moi aussi ce problème et je l'ai résolu de la manière suivante :

J'ai tout d'abord pensé à du papier de chocolat mais puisque ce "matériel" n'est pas auto-collant (seul le chocolat l'est...), j'ai eu l'idée de mettre des "gommettes" métallisées. C'est très économique, cela s'achète dans les librairies et c'est très facile à remplacer. J'espère que cela rendra service à tous les "Texans".

*Je voudrais aussi vous demander un tuyau. J'ai essayer de faire (en haute résolution) un graphique sur tout l'écran, mais mon dessin ne comportait que quelques "cases" identiques; tous les autres carrés avait un code ASCII différent. J'ai vite été à court de codes. Aussi, j'aimerais savoir s'il est possible de faire 768 (24*32) caractères différents et, si oui, comment.*

Jean-Pierre Diot - 47, Rue Jacquart - 51100 Reims

Nous vous remercions pour votre tuyau (et vos compliments) et nous espérons que l'article sur la haute résolution graphique, publié dans ce numéro, vous apportera quelques éclaircissements sur le sujet.

J'ai découvert récemment votre revue qui est une vraie mine d'or pour les utilisateurs du TI-99 puisqu'il est devenu extrêmement difficile de trouver, dans les magazines d'informatique individuelle, des programmes utilisables "tel quel", ou des idées intéressantes qui soient directement applicables au TI-99/4A. En effet, lorsqu'on débute dans cette discipline (dont on ignore tous les secrets) et que l'on essaie d'entrer un programmme non destiné au TI-99, on bute sur certaines instructions non prévues sur cet ordinateur, et le programme n'est pas exécutable, à cause de deux ou trois ligne mal adaptées.
J'aimerais savoir comment on peut remanier un programme quand on rencontre les instructions "POKE" ou "CLEAR" qui n'existent pas sur le TI, même en Basic Etendu.

Longue vie à votre revue !

François Palabaud - 5, Rue Grenette - 42190 Charlieu

Le Basic est un langage que l'on trouve sur tous les micro-ordinateurs, malheureusement, pratiquement chaque constructeur a son propre Basic, et l'adaptation des programmes n'est pas toujours chose aisée. L'instruction "POKE" est très courante et se trouve sur le TI sous le nom de "CALL LOAD", à condition de posséder le Basic Etendu, l'Editeur/Assembleur ou la Mini-mémoire. Cette instruction effectue le chargement d'une valeur directement dans une adresse de la mémoire vive. Donc, pour utiliser cette instuction dans une "traduction", il faut posséder une bonne connaissance de la structure des deux ordinateurs. En général, l'instruction "CLEAR" Provoque une réinitialisation (remise à zéro) de toutes les variables présentes dans le programme, et n'est pas disponible sur le TI-99. Par contre, il est parfaitement possible de la simuler :

- en Basic TI avec

```
150 A=0
160 B=0
170 C=0
etc...
```

- en Basic Etendu avec

```
150 A,B,C=0
etc...
```

Je tiens tout d'abord à vous féliciter et vous remercier pour la création d'un magazine destiné uniquement aux utilisateurs du TI-99 et la continuation de son édition malgré l'arrêt de fabrication de notre micro.
Il apparaît en effet qu'un magazine aussi spécialisé soit le seul investissement rentable pour l'utilisateur d'un micro, une fois son choix arrêté sur tel ou tel appareil. Je me réjouis donc que le TI-99 possède le sien.

J'en viens maintenant à mes questions :

- quels sont les éléments indispensables à l'accès au langage Assembleur ? D'après le manuel Basic Etendu, il semblerait que seule l'extension 32 Ko soit nécessaire. Est ce bien cela ?
- la brochure d'accompagnement du module Basic Etendu fait mention d'une option "Pre-scan" dont le but serait de supprimer l'analyse du programme après la commande "RUN", et d'éviter ainsi la perte de temps qui en résulte. Cependant, je ne retrouve pas cette option dans le manuel. Pouvez-vous m'indiquer le mode d'emploi de cette option ?

Pour terminer, je voudrais faire une ou deux remarques qui vous seront peut-être utiles.

Dans le numéro 2, Laurent Pecheux vous signalait une erreur dans le programme "Le dernier robot", en ligne 220. Il n'a en cela pas tout à fait tort. Il pourrait effectivement s'agir d'une erreur, mais d'une erreur par ommission. En effet, la ligne 220 a pour but de redéfinir les caractères après lecture en DATA du numéro et du descripteur du nouveau caractère. L'examen des lignes 5240 et 5250 fait ressortir que le programme redéfinit des caractères appartenant aux groupes 15 et 16 auxquels nous n'avons plus accès en Basic Etendu. Ainsi, si notre confrère a tenté de faire tourner le dernier robot en Basic Etendu, il est normal qu'il ait obtenu un message d'erreur et un

arrêt en ligne 220. Afin d'éviter ce genre de déboire, il serait utile que vous indiquiez, pour chaque programme en Basic TI, les incompatibilités avec le Basic Etendu, d'autant plus qu'elles sont relativement restreintes.

D'autre part, je pense qu'un article sur les erreurs, leurs messages et leur recherche pourrait être très intéressant pour les débutants. En effet, il n'y a rien de plus stressant qu'un programme qui bloque et de ne pouvoir y remédier. Il n'existe peut-être pas de recette pour rechercher une erreur, mais un petit truc comme l'utilisation de l'instruction "PRINT" permet très souvent de cerner la cause de l'erreur. Ainsi, Laurent Pecheux, une fois son programme bloqué en ligne 220, aurait pu tester la valeur des variables de cette ligne au moment du blocage par PRINT A et PRINT B$, et s'interroger sur l'exactitude et la possibilité des valeurs obtenues.

Christian Michel - 2, Rue F. Pelloutier - 38000 Grenoble

Pour accéder au langage Assembleur, il existe deux possibilités :

1) le module Editeur/Assembleur. Dans ce cas il faut disposer d'un lecteur de disquettes et de l'extension 32 Ko;

2) le module Mini-mémoire, avec lequel seule la console de base est nécessaire. L'assembleur ligne par ligne de la Mini-mémoire est beaucoup moins pratique que l'Editeur/Assembleur mais permet néanmoins de réaliser des petits programmes.

Il est en effet possible d'annuler partiellement l'action du "Pre-scan". La méthode à utiliser est indiquée dans le petit supplément fourni avec le manuel du Basic Etendu. Je ne dispose pas ici d'assez de place pour vous expliquer comment utiliser cette fonction. Nous pourrions développer cette particularité du Basic Etendu dans un prochain article mais, en attendant, vous pouvez consulter le programme de Gilbert Arribet baptisé "Sapin de Noël", publié dans le numéro 3, puisqu'il utilise cette fonction.

Nous avions mis en évidence, dans le courrier des lecteurs du numéro 3, les problèmes d'incompatibilité entre le Basic TI et le Basic Etendu, mais il n'était pas inutile d'y revenir. L'instruction "PRINT" est effectivement la plus efficace lorsqu'il s'agit de trouver la cause d'une erreur. Elle remplace avantageusement toutes les instructions de mise au point (TRACE, UNTRACE, etc...) dont l'utilisation est assez lourde.

Pouvez vous me dire si je peux me procurer votre magazine chez un marchand de journaux et, si oui, lequel ?

P. Denost - 35, rue Lamartine - 78000 Versailles

La revue "99 Magazine" est distribuée dans les boutiques informatiques et les librairies spécialisées par "PSI diffusion" mais n'est pas diffusée en kiosque.
D'autre part, vous êtes nombreux à vous être "cassé le nez" en essayant de venir nous voir, je profite donc de l'occasion pour signaler à nos lecteurs que le "49, rue Lamartine" est uniquement une adresse postale (en effet, les Editions MEV ne font, directement, que la vente par correspondance et le service des abonnements). ■

A vous de programmer

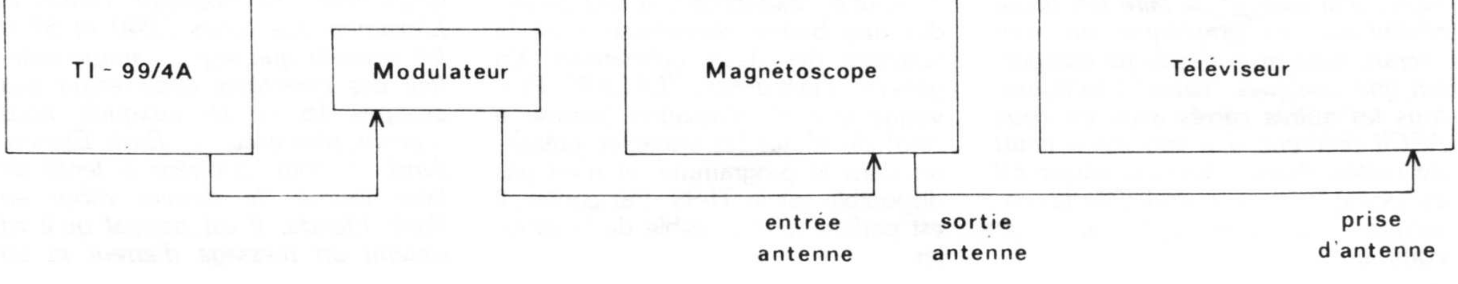
Si l'on en juge par la quantité de programmes simulant une horloge ou un chronomètre parvenus jusqu'à nos bureaux, il semble que cette rubrique soit appréciée.
Nous vous proposons donc à nouveau d'écrire un programme sur un thème donné. Comme par le passé, nous publierons le meilleur et son auteur recevra un abonnement pour quatre numéros avec cassettes.

Cette fois, il faudra écrire un programme ou un sous-programme capable de convertir des nombres entrés sous une forme numérique, en leurs équivalents "en toutes lettres". Par exemple :

1236861
deviendra :
UN MILLION DEUX CENTS TRENTE SIX MILLE HUIT CENTS SOIXANTE ET UN

Il est aussi envisageable de faire la conversion inverse, mais cela est un peu plus compliqué. Les lecteurs munis du Basic Etendu pourront créer un sous-programme à variables locales de manière à ce qu'il puisse être implanté dans n'importe quel autre logiciel.

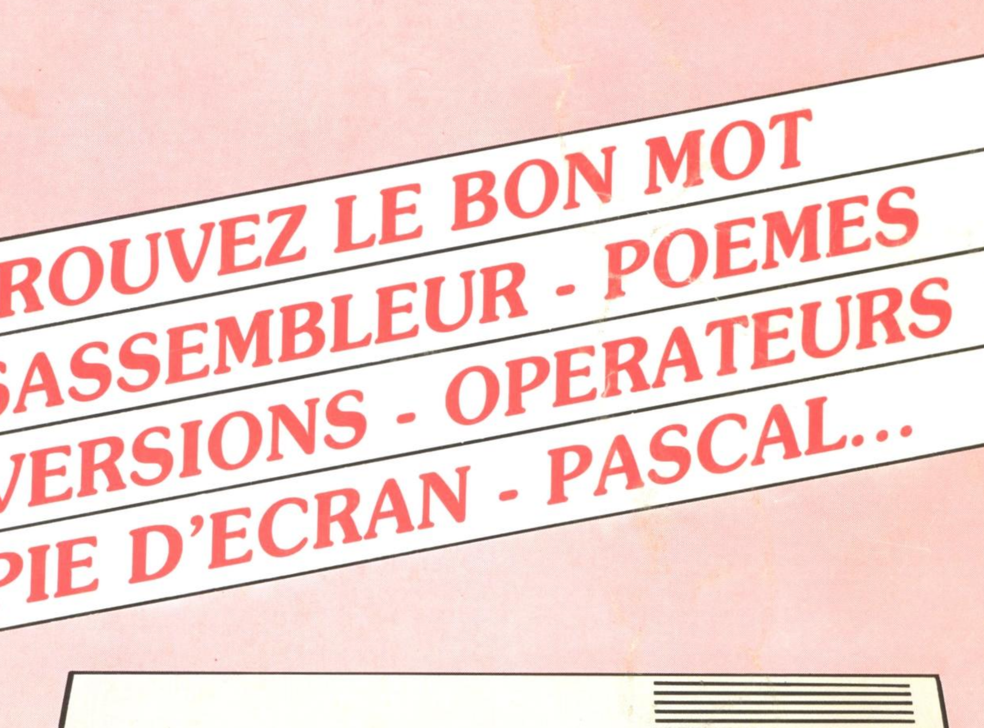
Le fruit de votre travail devra nous parvenir avant le premier mai, sauf pour nos amis Canadiens auxquels nous accordons deux semaines supplémentaires. Le temps nécessaire à l'acheminement du courrier entre la France et le Canada fait que notre ancienne méthode était un peu injuste. ■



99 MAGAZINE

CASSETTE D'ACCOMPAGNEMENT DISPONIBLE

TROUVEZ LE BON MOT
DESASSEMBLEUR - POEMES
CONVERSIONS - OPERATEURS
COPIE D'ECRAN - PASCAL...



NUMÉRO 5 - Trimestriel - Juin 1984 - 40 F

ISSN 0759-9005

99 MAGAZINE

Sommaire

	Page
Editorial par Hervé Thiriez	**5**
Conversions par Gilles Laurence	**6**
Systèmes linéaires par Alain Dupont	**10**
Structure des disques du P-System par Gérard Santraille	**15**
Poèmes et graphismes par Philippe Misandeau	**18**
Un désassembleur en Assembleur par Jean-Luc Bazanegue	**23**
A vous de programmer	**30**
Les tous de Hanoï par Daniel Masse	**31**
Trouvez le bon mot par Marianne Sutz	**33**
Le jeu de la vie par Gerbert Cinquet	**38**
Programmathèque par Marianne Sutz	**43**
Les opérateurs logiques par Julien Thomas	**44**
Apprendre le Basic en programmant par Roger Didi	**46**
Bibliographie par Sophie Ehster	**49**
Warangal par Edouard Krok	**50**
Graphisme en haute résolution par Gérard Santraille et Jean-Luc Bazanègue	**53**
Le trésor de l'Armada par Georges Herbepin	**56**
Ecriture littérale d'un nombre par Christine Julien	**58**
Copie d'écran sur imprimante Epson par Denise Amrouche	**59**
Description du port d'entrée/sortie par Eliane Dacquay	**60**
Translet par Georges Dejolier	**62**
Le club Ordinabis	**63**
Courrier des lecteurs par Alexandre Duback	**64**
Petites annonces	**67**
Le club Ticazur	**67**
Trucs et astuces	**32 - 49 - 52 - 63**

Les annonceurs
PSI Diffusion : page 2 - L'Ordinateur Individuel : page 68.

Editions MEV - 64-70, rue des Chantiers - 78000 Versailles

(Dépôt légal : 2e trimestre 1984)

Editorial

"99 magazine" a maintenant un an et, malgré les quelques "cafouillages" inévitables de nos débuts, il se porte bien.
La réorganisation de la rédaction, survenue juste après la parution du numéro 2, nous permet de vous proposer une revue susceptible d'intéresser tous les utilisateurs du TI-99/4A, du moins nous l'espérons.

Vous êtes encore nombreux à nous demander si nous continuerons à publier "99 Magazine". Vous devez donc savoir que la revue existera jusqu'à ce que nous n'ayons plus assez de lecteurs, ce qui est loin d'être le cas puisque vous êtes toujours plus nombreux à nous lire. Alors, abonnez-vous !

Dans ce numéro, encore très riche, ***Julien Thomas*** *poursuit sa série d'articles sur les particularités du basic Etendu avec, cette fois, les opérateurs logiques. Dans le même registre,* ***Gilles Laurence*** *vous propose un programme de conversion hexadécimal / binaire / décimal et, par la même occasion, passe en revue les différents systèmes de numération utilisés en informatique.*

Gérard Santraille *analyse la stucture des disques en Pascal et, en compagnie de* ***Jean-Luc Bazanegue****, ajoute une nouvelle routine graphique à votre TI.* ***Jean-Luc*** *a aussi écrit pour vous un désassembleur qui, s'il avait été commercialisé par Texas Instruments, vous aurait certainement coûté le prix d'un abonnement avec cassettes.*

Denise Amrouche *met à votre disposition deux programmes permettant de faire des copies d'écran sur imprimante, pendant que* ***Roger Didi*** *vous propose un nouvel article d'initiation au Basic.*

Vous pourrez transformer votre TI en poète grâce a ***Philippe Misandeau*** *qui vous présente un programme original, d'une façon non moins originale. Le programme "Systèmes linéaires", écrit par* ***Alain dupont****, démontre que notre ordinateur est une excellente machine de calcul pendant que* ***Gerbert Cinquet*** *présente deux "jeu de la vie"; le premier est en Basic et le second en Assembleur.*

"Trouvez le bon mot" est un programme dérivé du Mastermind et écrit par ***Marianne Sutz****, qui s'est aussi chargé de la rubrique "programmathèque".* ***Edouard Krok*** *et* ***Daniel Masse*** *apportent à nouveau leur contribution sous la forme des programmes "Warangal" et "Les tours de Hanoï".* ***Georges Herbepin*** *nous propose lui aussi un très bon programme de jeu intitulé "Le trésor de l'Armada"*

Nous avons cette fois deux gagnants à notre petit jeux "A vous de programmer" : ***Christine Julien*** *pour un programme en Pascal, et* ***Georges Dejolier*** *pour un programme en Basic Etendu baptisé "Translet".*

Les réponses au courrier des lecteurs sont toujours assurées par ***Alexandre Duback****, tandis que* ***Sophie Ehster*** *prend en main la rubrique "Bibliographie.*

*Les amateurs de "hardware" ne sont pas laissés pour compte puisqu'****Eliane Dacquay*** *vous donne la description du port d'entrées/sorties de notre cher appareil. Comme à l'accoutumée, vous trouverez aussi dans ce numéro de nombreux trucs et astuces.*

Nous vous souhaitons de bonnes vacances et vous donnons rendez-vous en Septembre, au Sicob.

Hervé Thiriez

Rédacteur en chef-Directeur de la publication : Hervé Thiriez. **Comité de rédaction :** Jean-Luc Bazanegue, Roger Didi, Gérard Santraille. **Dessins :** Laurent Bidot. **Ont collaboré à ce numéro :** Denise Amrouche, Gérard Baroni, Jean-Luc Bazanegue, Omar Belmokar, Gerbert Cinquet, Eliane Dacquay, Georges Dejolier, Roger Didi, Alexandre Duback, Alain Dupont, Sophie Ehster, Georges Herbepin, Antoine Heudes, Christine Julien, Edouard Krok, Gilles Laurence, Jean Marlin, Daniel Masse, Philippe Misandeau, Gérard Santraille, Marianne Sutz, Julien Thomas, Pascal Thomas, Marc Roch-Vilato.

Editions MEV - 64-70, rue des Chantiers - 78000 Versailles.

Régie publicitaire : Force 7 - Anne Jourdan - 5, place du Colonel Fabien, 75010 Paris - Tél. : 240.22.01.

Diffusion auprès des boutiques et librairies : PSI Diffusion, BP 86, 77402 Lagny-sur-Marne Cedex.

Composition : Télécompo - 13-15, avenue du Petit Parc, 94300 Vincennes.

Impression : Imprimerie Rosay - 47, avenue de Paris, 94300 Vincennes - Tél. : (1) 328.18.63.

Conversions

Gilles Laurence

Les différents systèmes de numération utilisés en informatique ne sont pas encore parfaitement maîtrisés par tous les lecteurs de "99 Magazine". Il semblait donc nécessaire d'écrire un petit article expliquant comment fonctionnent ces systèmes, le programme de conversion décimal / hexadécimal / binaire en **Basic TI** listé ci-après n'étant qu'un prétexte.

Le système décimal (base 10)

D'aucuns se demanderont certainement quelle est la raison qui nous pousse à vouloir expliquer un système de numération utilisé quotidiennement depuis notre plus tendre enfance. Ceci ne sera pas inutile car nous nous apercevrons que *tous* les systèmes de numération fonctionnent de la même façon.

Prenons un nombre exprimé en décimal, 147908 par exemple. Ce nombre doit être décomposé en tranches, et ce de la droite vers la gauche. Cela nous donne la série de chiffres 8 - 0 - 9 - 7 - 4 - 1. Ici intervient quelque chose de fondamental : la notion de *poids ou de rang*.
Par convention, les chiffres constituant le nombre sont numérotés de 0 à N, N étant le numéro du chiffre placé le plus à droite. Ces numéros représentent le poids ou rang des chiffres; dans notre exemple, ces rangs sont les suivants :

8 : rang 0 (poids le plus faible)
0 : rang 1
9 : rang 2
7 : rang 3
4 : rang 4
1 : rang 5 (poids le plus fort du nombre)

Maintenant, pour obtenir la valeur réelle de notre nombre, il faut multiplier chaque chiffre par la base (10) élevée à la puissance X, X étant le rang (ou poids) du chiffre considéré. L'addition des résultats de ces opérations nous donne la valeur recherchée. Reprenons notre exemple :

- 8*(10^0) = 8*1 = 8 (10^0 est égal à 1)
- 0*(10^1) = 0*10 = 0
- 9*(10^2) = 9*100 = 900
- 7*(10^3) = 7*1000 = 7000
- 4*(10^4) = 4*10000 =40000
- 1*(10^5) = 1*100000 = 100000
- 8 + 0 + 900 + 7000 + 40000 + 100000 = 147908

Nous retrouvons bien notre nombre.

Le principe de lecture du nombre de la droite vers la gauche peut paraître curieux, mais il faut savoir que les systèmes de numération par "tranches" nous viennent des mathématiciens arabes qui, eux, écrivaient de droite à gauche

Le système hexadécimal (base 16)

En décimal, chaque position dans le nombre peut contenir les chiffres 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9. Lorsqu'on effectue une addition, si le résultat est supérieur à 9, il y a retenue au rang supérieur. Par exemple, si nous additionnons 8 et 4, la somme ne peut pas être représentée dans un seul rang et il y a une retenue :

8 + 4 = 12 soit 2 au rang 0 et 1 au rang 1.

En hexadécimal, le principe est exactement le même mais un rang peut contenir des valeurs comprises entre 0 et 15 et les retenues n'apparaissent qu'à partir de 16.
Comme on ne connaît habituellement que 10 chiffres (0 à 9), les valeurs 10 à 15 sont représentées par les six premières lettres de l'alphabet :
A=10, B=11, C=12, D=13, E=14 et F=15.

Ainsi, si l'on additionne 9 et 4 (en hexadécimal), nous obtenons D au rang 0 et aucune retenue puisque 9+4 est égal à 13 en décimal, soit D en hexadécimal. Si nous additionnons C et D, le résultat est 19 (attention, nous ne sommes plus en décimal). Une petite explication est peut-être souhaitée :

C est égal à 12 en décimal et D est égal à 13; le produit de l'addition est 25, donc supérieur à 16, d'où la nécessité de prendre une retenue en compte;
nous allons décomposer le nombre que l'on ajoute à C, soit D=13 en décimal, en deux parties, dont l'une correspond à la valeur qu'il faut ajouter à C (12 en décimal) pour arriver à 16 et l'autre à la différence *entre D et cette valeur;*
en ajoutant 4 à 12, on obtient 16, soit 10 en hexadécimal (1*(16^1) + 0*(16^0)). Sur le nombre 13 que nous devions ajouter, il nous reste donc 9. Le résultat final de l'opération vaut par conséquent 10+9 = 19 en héxadécimal.

En résumé : C + D = 12 + 13 = 12 +4 + 9 = (12 + 4) + 9 = 16 + 9 en décimal = (1*16) + 9 en décimal = 19 en hexadécimal (1 dans le chiffre des "seizaines" et 9 dans le chiffre des unités).

Essayons maintenant de convertir le nombre F8B2 (hexadécimal) en son équivalent en base 10 :

- F est au rang 3, ce qui nous donne 15*(16^3) = 15*4096 = 61440;
- 8 est au rang 2, 8*(16^2) = 8*256 = 2048;
- B est au rang 1, 11*(16^1) = 11*16 = 176;
- 2 est au rang 2, 2*(16^0) = 2*1 = 2.

Si l'on additionne les résultats de ces quatre opérations, on obtient 61440 + 2048 + 176 + 2 soit 63666, par conséquent, F8B2 est égal à 63666 en décimal.

Le système binaire (Base 2)

Le système de numération en base 2 est le seul qui soit reconnu par les microprocesseurs. En fait, si nous allions un peu plus loin, nous nous apercevrions qu'un tel circuit ne peut *utiliser que des différences de potentiel*, mais là n'est pas le propos de cet article.

Un petit problème : que font 1 et 1 en binaire ? Ceux qui ont répondu 2 se sont trompés. En effet, si les retenues n'apparaissent qu'à partir de 10 en décimal et 16 en hexadécimal, elles sont requises dès la valeur 2 en *binaire, puisque les seuls chiffres* utilisables dans ce système sont 0 et 1. Les additions en binaire s'effectuent de la manière suivante :

0+0 = 0, pas de retenue;
0+1 = 1, pas de retenue;
1+1 = 0 avec retenue de 1;
1+1+retenue = 1 avec retenue de 1.

1+1 en binaire *est* donc égal à 10 ou, en décimal, à 1*(2^1) + 0*(2^0) = 1*2 + 0*1 = 2.

Pour faire la conversion d'un nombre *en base 2 en son équivalent décimal*, il suffit d'appliquer les règles utilisées pour les autres systèmes de numération. Prenons par exemple le nombre binaire 10101110 composé de 8 chiffres appelés dans ce cas "bits" (pour Binary digIT soit élément binaire) :

- rang 7 = 1, donc 1*(2^7) = 128;

- rang 6 = 0, 0★(2ˆ6) = 0;
- rang 5 = 1, 1★(2ˆ5) = 32;
- rang 4 = 0, 0★(2ˆ4) = 0;
- rang 3 = 1, 1★(2ˆ3) = 8;
- rang 2 = 1, 1★(2ˆ2) = 4;
- rang 1 = 1, 1★(2ˆ1) = 2;
- rang 0 = 0, 0★(2ˆ0) = 0.
- 10101110 est égal à 128 + 0 + 32 + 0 + 8 + 4 + 2 + 0 = 174.

Le bit situé au rang 0 ne peut représenter que les valeurs 0 et 1. Cette particularité peut être utilisée pour déterminer si l'on a affaire à un nombre pair ou impair.

Le complément à deux

Il y a deux manières d'utiliser les nombres exprimés en base 2 : en valeur absolue (c'est le cas exposé au paragraphe précédent) ou en valeur signée. Cette dernière méthode permet de faire la différence entre un nombre positif et un nombre négatif.

Pour une utilisation correcte du complément à deux, il est indispensable de déterminer le nombre de bits utilisés pour la représentation du nombre. En général, la taille utilisée est l'octet (8 bits) ou le mot (16 bits). Nous retiendrons la dernière possibilité puisque le microprocesseur du TI-99 traite des mots de 16 bits.
En complément à deux, le bit de poids fort (rang 15 pour un mot de 16 bits) indique le signe du nombre : 1 pour un nombre négatif et 0 pour un nombre positif. Ainsi, 1000000000000001 est négatif alors que 0000000000000001 est positif.

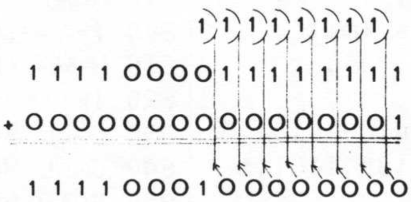
Pour obtenir la représentation en complément à deux d'un nombre négatif, il faut d'abord convertir la valeur absolue de ce nombre en binaire. Par exemple, la valeur décimale −3840 devient 3840, soit en binaire 0000111100000000 (2048 + 1024 + 512 +256). Ensuite, deux étapes doivent être respectées :

1) le complément à un

Cette opération est extrêmement simple à réaliser; il suffit de savoir que le complément à un de 1 est 0 et le complément à un de 0 est 1. Cela revient en fait à inverser la valeur de tous les bits constituant le nombre binaire à traiter : le nombre 0000111100000000 devient, en complément à un, 11110000111111 11.

2) ajout de 1 au résultat précédent

Pour obtenir le complément à deux du nombre il faut maintenant ajouter 1 (0000000000000001 en binaire) au complément à un. Si vous vous souvenez des règles utilisées pour les additions en binaire (paragraphe précédent), cette opération ne présente pas de difficulté. Sinon, le schéma suivant peut certainement vous éclairer :



Le complément à deux de 0000111100000000 (2048 + 1024 + 512 + 256 = 3840) est donc 1111000100000000 ou, en décimal, 16384 + 8192 + 4096 + 256 = −28928 (le bit 15 est à 1, donc la valeur est négative et seuls les bits 0 à 14 sont pris en compte pour la représentation du nombre).

Nous pouvons noter au passage qu'un nombre binaire de 16 bits peut représenter, en valeur absolue, toutes les valeurs comprises entre 0 et 65535 (2ˆ16−1), alors qu'en complément à deux les valeurs possibles sont comprises entre −32768 et 32767 (2ˆ15 −1).

Octal (base 8) et Décimal codé binaire

Ces deux systèmes de numération ne sont pas utilisés sur le TI-99 mais, comme nous parlons des différents modes de représentation des nombres, autant passer ceux-ci en revue.

L'octal

Ce type de numération était autrefois fréquemment utilisé en informatique, surtout sur les gros systèmes, mais il a actuellement tendance à disparaître.
La base employée est 8; il y a donc retenue seulement à partir de 8 et les chiffres utilisés sont 0, 1, 2, 3, 4, 5, 6 et 7. Les règles sont les mêmes que pour les autres systèmes de numération.

Le décimal codé binaire (DCB)

Ce système est très pratique lorsque l'on veut effectuer des opérations sur des valeurs décimales, mais en binaire. Pour cela, chaque chiffre utilisé en décimal est codé en binaire sur 4 bits (quartet). Les codes sont les suivants :
0000 = 0
0001 = 1
0010 = 2
0011 = 3
0100 = 4
0101 = 5
0110 = 6
0111 = 7
1000 = 8
1001 = 9
La conversion décimal / DCB s'effectue très facilement puisque, par exemple, 1234 en décimal devient 0001 0010 0011 0100 en décimal codé binaire.

Le programme

Ce programme de conversion est écrit en **Basic TI** mais il fonctionne aussi en **Basic Etendu**, où les résultats sont obtenus un peu plus rapidement. Pour entrer un nombre à convertir, il suffit d'indiquer au programme le système de numération employé, par exemple en appuyant sur la touche "B" pour un nombre binaire. Vous pouvez alors entrer votre nombre.

D)écimal

Les nombres décimaux doivent être compris entre −32768 et 65535; si tel n'est pas le cas, le nombre est rejeté. Le programme vous donne l'équivalent de la valeur en hexadécimal et en binaire sur 16 bits.

H)exadécimal

Seuls les caractères 0 à 9 et A à F sont autorisés. Il est possible de communiquer au programme tout nombre hexadécimal compris entre 0000 et FFFF. Vous obtenez le résultat de la conversion en binaire sur 16 bits et en décimal. Si nécessaire, la valeur en complément à deux sera aussi fournie (sur 8 bits si cela est possible, sinon sur 16 bits).

B)inaire

Seuls les chiffres 0 et 1 sont valides et le programme accepte jusqu'à 16 bits. Les valeurs en hexadécimal et en décimal sont affichées ainsi que, si nécessaire, le complément à deux du nombre. ■

```
100 REM ********************
110 REM *    CONVERSIONS    *
120 REM * deci/binaire/hexa *
130 REM ********************
140 REM *     Basic  TI     *
150 REM ********************
160 REM
170 CALL CLEAR
```

```
180 CALL SCREEN(8)
190 DIM B$(15)
200 FOR B=0 TO 15
210 READ B$(B)
220 NEXT B
230 P$=" ***************************"
240 PRINT P$:TAB(10);"CONVERSIONS":" deci
mal/binaire/hexadecimal":P$: :
250 PRINT "    B)inaire    D)ecimal":"
H)exadecimal    A)rret": : : : : : : :
: : : : : : : : :
260 DATA 0000,0001,0010,0011,0100,0101,01
10,0111,1000,1001,1010,1011,1100,1101,1
110,1111
270 DE$="Decimal : "
280 HE$="Hexadecimal : >"
290 BIN$="Binaire : "
300 VA$="valeur abs"
310 C2$="compl. a 2"
320 REM
330 REM saisie base depart
340 REM
350 CR=0
360 CALL KEY(0,K,S)
370 IF S THEN 470
380 CR=CR+1
390 IF CR<3 THEN 360
400 IF DR THEN 440
410 DR=1
420 CALL HCHAR(9,4,95)
430 GOTO 350
440 DR=0
450 CALL HCHAR(9,4,32)
460 GOTO 350
470 IF K=65 THEN 3100
480 IF K=66 THEN 1110
490 IF K=72 THEN 2030
500 IF K<>68 THEN 360
510 REM
520 REM saisie nombre decimal
530 REM
540 T$=DE$
550 L=9
560 C=4
570 GOSUB 2870
580 C$=""
590 FOR C=14 TO 19
600 CR=0
610 CALL KEY(0,K,S)
620 IF S=1 THEN 730
630 CR=CR+1
640 IF CR<3 THEN 610
650 IF DR THEN 690
660 DR=1
670 CALL HCHAR(L,C,95)
680 GOTO 600
690 DR=0
700 CALL HCHAR(L,C,32)
710 GOTO 600
720 IF S<1 THEN 630
730 IF K=13 THEN 800
740 IF K<>45 THEN 760
750 IF C=14 THEN 770
760 IF (K<48)+(K>57)THEN 610
770 CALL HCHAR(L,C,K)
780 C$=C$&CHR$(K)
790 NEXT C
800 IF C=14 THEN 580
810 N=VAL(C$)
820 IF (N>-32769)*(N<65536)THEN 860
830 CALL SOUND(5,1000,0)
840 CALL HCHAR(L,14,32,6)
850 GOTO 580
860 T$=C$&" "
870 C=14
880 GOSUB 2870
890 IF N>-1 THEN 910
900 N=N+65536
910 GOSUB 2730
920 GOSUB 2610
930 N=VAL(C$)
940 IF (N>256)+(N<-255)THEN 980
950 H$=SEG$(H$,3,2)
960 BI$=SEG$(BI$,9,8)
970 GOTO 990
980 BI$=SEG$(BI$,1,8)&" "&SEG$(BI$,9,8)
990 T$=HE$&H$
1000 L=11
1010 C=4
1020 GOSUB 2870
1030 T$=BIN$&BI$
1040 L=13
1050 GOSUB 2870
1060 GOSUB 2960
1070 GOTO 380
1080 REM
1090 REM saisie nombre binaire
1100 REM
1110 T$=BIN$
1120 L=9
1130 C=4
1140 GOSUB 2870
1150 C$=""
1160 FOR C=14 TO 29
1170 CR=0
1180 CALL KEY(0,K,S)
1190 IF S=1 THEN 1290
1200 CR=CR+1
1210 IF CR<3 THEN 1180
1220 IF DR THEN 1260
1230 DR=1
1240 CALL HCHAR(L,C,95)
1250 GOTO 1170
1260 DR=0
1270 CALL HCHAR(L,C,32)
1280 GOTO 1170
1290 IF K=13 THEN 1340
1300 IF (K<48)+(K>49)THEN 1180
```

```
1310 CALL HCHAR(L,C,K)
1320 C$=C$&CHR$(K)
1330 NEXT C
1340 IF C=14 THEN 1150
1350 LO=LEN(C$)+1
1360 FOR B=LO TO 16
1370 C$="0"&C$
1380 NEXT B
1390 T$=SEG$(C$,1,8)&" "&SEG$(C$,9,8)
1400 C=14
1410 GOSUB 2870
1420 GOSUB 2520
1430 ND=N
1440 GOSUB 2730
1450 IF ND>255 THEN 1750
1460 H$=SEG$(H$,3,2)
1470 T$=HE$&H$
1480 L=11
1490 C=4
1500 GOSUB 2870
1510 REM
1520 REM partie commune
1530 REM hexa/binaire
1540 REM
1550 IF ND<128 THEN 1710
1560 N=ND-256
1570 T$=DE$&VA$
1580 L=13
1590 GOSUB 2870
1600 T$=STR$(ND)
1610 C=30-LEN(T$)
1620 GOSUB 2870
1630 T$=C2$
1640 L=14
1650 C=14
1660 GOSUB 2870
1670 T$=STR$(N)
1680 C=30-LEN(T$)
1690 GOSUB 2870
1700 GOTO 1980
1710 T$=DE$&STR$(ND)
1720 L=13
1730 GOSUB 2870
1740 GOTO 1980
1750 T$=HE$&H$
1760 L=11
1770 C=4
1780 GOSUB 2870
1790 IF ND<32768 THEN 1950
1800 N=ND-65536
1810 T$=DE$&VA$
1820 L=13
1830 GOSUB 2870
1840 T$=STR$(ND)
1850 C=31-LEN(T$)
1860 GOSUB 2870
1870 T$=C2$
1880 L=14
1890 C=14
1900 GOSUB 2870
1910 T$=STR$(N)
1920 C=31-LEN(T$)
1930 GOSUB 2870
1940 GOTO 1980
1950 T$=DE$&STR$(ND)
1960 L=13
1970 GOSUB 2870
1980 GOSUB 2960
1990 GOTO 380
2000 REM
2010 REM saisie nombre hexadecimal
2020 REM
2030 T$=HE$
2040 L=9
2050 C=4
2060 GOSUB 2870
2070 C$=""
2080 FOR C=19 TO 22
2090 CR=0
2100 CALL KEY(0,K,S)
2110 IF S=1 THEN 2210
2120 CR=CR+1
2130 IF CR<3 THEN 2100
2140 IF DR THEN 2180
2150 DR=1
2160 CALL HCHAR(L,C,95)
2170 GOTO 2090
2180 DR=0
2190 CALL HCHAR(L,C,32)
2200 GOTO 2090
2210 IF K=13 THEN 2260
2220 IF (K<48)+(K>57)*(K<65)+(K>70)THEN 2
100
2230 CALL HCHAR(L,C,K)
2240 C$=C$&CHR$(K)
2250 NEXT C
2260 IF C=19 THEN 2070
2270 LO=LEN(C$)+1
2280 FOR B=LO TO 4
2290 C$="0"&C$
2300 NEXT B
2310 T$=C$&" "
2320 C=19
2330 GOSUB 2870
2340 H$=C$
2350 GOSUB 2610
2360 C$=BI$
2370 GOSUB 2520
2380 IF (N>256)+(N<-255)THEN 2410
2390 BI$=SEG$(BI$,9,8)
2400 GOTO 2420
2410 BI$=SEG$(BI$,1,8)&" "&SEG$(BI$,9,8)
2420 T$=BIN$&BI$
2430 L=11
2440 C=4
2450 GOSUB 2870
2460 ND=N
2470 IF ND>255 THEN 1790
```

```
2480 GOTO 1550
2490 REM
2500 REM SP binaire->decimal
2510 REM
2520 N=0
2530 FOR B=15 TO 0 STEP -1
2540 IF SEG$(C$,16-B,1)="0" THEN 2560
2550 N=N+2^B
2560 NEXT B
2570 RETURN
2580 REM
2590 REM SP hexa->binaire
2600 REM
2610 BI$=""
2620 FOR B=1 TO 4
2630 X=ASC(SEG$(H$,B,1))
2640 IF X>57 THEN 2670
2650 BI$=BI$&B$(X-48)
2660 GOTO 2680
2670 BI$=BI$&B$(X-55)
2680 NEXT B
2690 RETURN
2700 REM
2710 REM SP decimal->hexa
2720 REM
2730 H$=""
2740 FOR B=3 TO 0 STEP -1
2750 X=2^(B*4)
2760 NI=INT(N/X)
2770 N=N-NI*X
2780 IF NI<10 THEN 2810
2790 H$=H$&CHR$(NI+55)
2800 GOTO 2820
2810 H$=H$&CHR$(NI+48)
2820 NEXT B
2830 RETURN
2840 REM
2850 REM SP "display at"
2860 REM
2870 PC=1
2880 FOR B=C TO C+LEN(T$)-1
2890 CALL HCHAR(L,B,ASC(SEG$(T$,PC,1)))
2900 PC=PC+1
2910 NEXT B
2920 RETURN
2930 REM
2940 REM SP "pour continuer"
2950 REM
2960 T$="<ENTER> pour continuer..."
2970 C=4
2980 L=24
2990 GOSUB 2870
3000 CALL KEY(0,K,S)
3010 IF K<>13 THEN 3000
3020 FOR B=9 TO 14
3030 CALL HCHAR(B,4,32,27)
3040 NEXT B
3050 CALL HCHAR(24,4,32,27)
3060 RETURN
3070 REM
3080 REM arret programme
3090 REM
3100 CALL CLEAR
3110 END
```

Systèmes linéaires

Alain Dupont

Ce programme écrit en **Basic TI** permet la résolution d'un système de N équations linéaires à N inconnues. N est au plus égal à 10, mais il est facile d'étendre le programme à un nombre supérieur d'équations.

Le principe

On utilise ici la méthode du pivot de Gauss dont le fonctionnement est illustré sur l'exemple ci-dessous :

soit à résoudre le système suivant

```
A11.X1 + A12.X2 + A13.X3 = A14
A21.X1 + A22.X2 + A23.X3 = A24
A31.X1 + A32.X2 + A33.X3 = A34
```

X1, X2, X3 sont les inconnues; les termes Aij sont des coefficients. A partir de ce système, on construit un système équivalent, plus facile à résoudre :

```
A11.X1 + A12.X2 + A13.X3 = A14
         B22.X2 + B23.X3 = B24
                  C33.X3 = C34
```

On a alors :
X3 = C34/C33
X2 = (B24−B23.X3)/B22
X1 = (A14−A12.X2−A13.X3)/A11

Tout le travail consiste à exprimer les coefficients Bij et Cij en fonction des Aij. Le programme manipule des lignes de coefficients à partir du tableau initial :

```
/ A11 A12 A13 ! A14 \    Ligne L1
!                   !
! A21 A22 A23 ! A24 !    Ligne L2
!                   !
\ A31 A32 A33 ! A34 /    Ligne L3
```

Dans le but d'annuler B21 et C31, on réalise les manipulations suivantes :
la ligne L2 est remplacée par L2−(A21/A11).L1;
la ligne L3 est remplacée par L3−(A31/A11).L1

On dit qu'on utilise ici le pivot A11; on obtient alors le tableau :

```
/ A11 A12 A13 ! A14 \    Ligne L1
!                   !
!  0  B22 B23 ! B24 !    Ligne L2
!                   !
\  0  E32 E33 ! E34 /    Ligne L3
```

Pour annuler E32, la ligne L3 est remplacée par L3−(E32/B22).L2; on utilise ici le pivot B22 et on obtient le tableau final :

```
/ A11 A12 A13 ! A14 \
!                   !
!  0  B22 B23 ! B24 !
!                   !
\  0   0  C33 ! C34 /
```

En cas de pivot nul, on permute la ligne concernée avec la ligne suivante. Pour de plus amples informations sur la méthode du pivot de Gauss, vous pouvez consulter un ouvrage de mathématiques du niveau de la classe de terminale C.

Le programme

Il est écrit en Basic TI et occupe environ 10 Ko (voir "99 magazine" numéro 3, page 9 pour l'évaluation de la taille du programme).

Pour l'affichage, l'écran est rendu transparent (s.p. 3510) pendant le défilement vertical des caractères, puis redevient normal ensuite (s.p. 3560). Les routines utilisées ont été placées en sous-programmes :

2390 : attente d'une pression sur une touche;
2450 : temporisation;
2490, 2520, 2590, 2690 : utilisation de la ligne 24 à l'écran;
2200 : permutation des lignes en cas de pivot nul. Il se peut que le pivot reste nul malgré tout; dans ce cas, on affecte la valeur 2 à la variable BLOC; dans le cas contraire, la valeur 1. L'état de BLOC est testé après chaque exécution du sous-programme 2200. Si BLOC=2, cela signifie que le système n'a pas de solution unique (il y en a une infinité ou aucune);
2780 à 3450 : série de sous-programmes dont le rôle est de diagonaliser le tableau des coefficients comme indiqué précédemment dans le cas de 3 inconnues; 2780 correspond à l'utilisation du pivot A11, 2850 correspond à A22, et ainsi de suite;
3610 : calcul des solutions X1, X2,...Xn;
3720 : affichage des solutions.

La manipulation du tableau des coefficients ne commence qu'en ligne 900. Auparavant, le TI-99 présente le programme, demande le nombre d'équations à résoudre, puis demande l'introduction des coefficients ligne par ligne (110 coefficients si vous voulez résoudre un système de 10 équations à 10 inconnues !); ne pas oublier les coefficients nuls.

Après chargement, vous pouvez modifier tout coefficient si vous le désirez. La ligne 930 est d'une importance capitale : elle dirige le programme en fonction du nombre d'équations à résoudre (940 : 2 équations, 1220 : 3 équations, etc... jusqu'à 2030 : 10 équations).

En début de programme, on remarquera plusieurs tableaux (ligne 130) : le tableau A(10,11) sert à stocker les coefficients; le tableau NA(11) sert de relais pour échanger deux lignes de coefficients; quant au tableau X(11), il sert à stocker les solutions.

Ce programme vous impressionnera par sa puissance de calcul, et démontrera si besoin est que le TI-99 peut être une machine de calcul efficace (essayez de résoudre un système vous-même et comparez votre temps de calcul avec celui de la bête !). ■

```
10 REM ********************
20 REM *     Systemes      *
30 REM *     Lineaires     *
40 REM ********************
50 REM *     Basic  TI     *
60 REM ********************
70 REM
100 CALL CLEAR
110 CALL SCREEN(13)
120 OPTION BASE 1
130 DIM A(10,11),NA(11),X(11)
140 U=0
150 GOSUB 3510
160 GOSUB 2350
170 GOSUB 3560
180 GOSUB 2390
190 CALL CLEAR
200 GOSUB 3510
210 GOSUB 3780
220 GOSUB 3560
230 GOSUB 2390
240 CALL CLEAR
250 GOSUB 3510
260 GOSUB 3880
270 GOSUB 3560
280 GOSUB 2390
290 GOSUB 400
300 GOSUB 2390
310 CALL CLEAR
320 INPUT "nombre d'equations?  ":N
330 IF (N>=2)*(N<=10)THEN 510 ELSE 340
340 CALL CLEAR
350 PRINT "nombre non admis par le progra
mme"
360 FOR DEL=1 TO 500
370 NEXT DEL
380 CALL CLEAR
390 GOTO 320
400 CALL CLEAR
410 GOSUB 3510
420 PRINT "le systeme est de la forme": :
 : :
430 PRINT "A11X1+A12X2+..+A1NXN=A1,N+1":
:
440 PRINT "A21X1+A22X2+..+A2NXN=A2,N+1":
:
450 PRINT "..........................": :
460 PRINT "AN1X1+AN2X2+..+ANNXN=AN,N+1":
: : : :
470 PRINT "introduire les coefficients":
:"AIJ ligne par ligne": : :
480 GOSUB 3560
490 RETURN
500 GOSUB 2390
510 CALL CLEAR
520 FOR I=1 TO N
530 FOR J=1 TO N+1
540 PRINT "introduisez  ";"A";I;J: : : :
:
550 INPUT A(I,J)
560 CALL CLEAR
570 NEXT J
580 NEXT I
590 PRINT "introduction terminee"
600 GOSUB 2450
610 CALL CLEAR
620 GOSUB 3510
630 PRINT "voulez vous modifier des": :"c
oefficients? (O/N)": : : : : : : :
640 GOSUB 3560
650 CALL SOUND(200,500,0)
660 CALL SOUND(200,800,0)
670 CALL KEY(0,KK,SS)
680 IF SS=0 THEN 670
690 IF (KK<>79)*(KK<>78)THEN 670
700 IF KK=79 THEN 710 ELSE 900
710 GOSUB 2490
```

```
720 GOSUB 2520
730 GOSUB 2590
740 CALL KEY(0,KI,SI)
750 IF SI=0 THEN 740
760 CALL HCHAR(24,18,KI)
770 GOSUB 2450
780 GOSUB 2490
790 GOSUB 2520
800 GOSUB 2690
810 CALL KEY(0,KJ,SJ)
820 IF SJ=0 THEN 810
830 CALL HCHAR(24,18,KJ)
840 GOSUB 2450
850 CALL CLEAR
860 PRINT "introduisez le coefficient": :
"A";KI-48;KJ-48: : :
870 PRINT "valeur precedente: ";A(KI-48,K
J-48): : :
880 INPUT "nouvelle valeur:?   ":A(KI-48,
KJ-48)
890 GOTO 610
900 REM *resolution du systeme*
910 CALL CLEAR
920 PRINT "resolution en cours": : : :
930 ON N-1 GOTO 940,1220,1340,1430,1530,1
640,1760,1890,2030
940 REM *2 equations*
950 B=1
960 GOSUB 2200
970 IF BLOC=2 THEN 3940
980 GOSUB 2780
990 IF A(N,N)=0 THEN 3810
1000 GOSUB 3610
1010 GOSUB 3720
1020 GOSUB 2390
1030 CALL CLEAR
1040 PRINT "une autre resolution? (O/N) "
: : : :
1050 CALL SOUND(500,700,0,900,1)
1060 CALL KEY(0,K,S)
1070 IF S=0 THEN 1060
1080 IF (K<>79)*(K<>78)THEN 1060
1090 IF K=79 THEN 1100 ELSE 2130
1100 CALL CLEAR
1110 PRINT "OK pour une autre resolution"
: :
1120 CALL SOUND(500,700,0,900,1)
1130 FOR I=1 TO 10
1140 FOR J=1 TO 11
1150 X(J)=0
1160 NA(J)=0
1170 A(I,J)=0
1180 NEXT J
1190 NEXT I
1200 CALL CLEAR
1210 GOTO 320
1220 REM *cas de 3 equations*
1230 GOSUB 1250
1240 GOTO 990
1250 B=1
1260 GOSUB 2200
1270 IF BLOC=2 THEN 3940
1280 GOSUB 2780
1290 B=2
1300 GOSUB 2200
1310 IF BLOC=2 THEN 3940
1320 GOSUB 2850
1330 RETURN
1340 REM * cas de 4 equations*
1350 GOSUB 1250
1360 GOSUB 1380
1370 GOTO 990
1380 B=3
1390 GOSUB 2200
1400 IF BLOC=2 THEN 3940
1410 GOSUB 2920
1420 RETURN
1430 REM *cas de 5 equations *
1440 GOSUB 1250
1450 GOSUB 1380
1460 GOSUB 1480
1470 GOTO 990
1480 B=4
1490 GOSUB 2200
1500 IF BLOC=2 THEN 3940
1510 GOSUB 3010
1520 RETURN
1530 REM *cas de 6 equations*
1540 GOSUB 1250
1550 GOSUB 1380
1560 GOSUB 1480
1570 GOSUB 1590
1580 GOTO 990
1590 B=5
1600 GOSUB 2200
1610 IF BLOC=2 THEN 3940
1620 GOSUB 3090
1630 RETURN
1640 REM *cas de 7 equations*
1650 GOSUB 1250
1660 GOSUB 1380
1670 GOSUB 1480
1680 GOSUB 1590
1690 GOSUB 1710
1700 GOTO 990
1710 B=6
1720 GOSUB 2200
1730 IF BLOC=2 THEN 3940
1740 GOSUB 3170
1750 RETURN
1760 REM *cas de 8 equations*
1770 GOSUB 1250
1780 GOSUB 1380
1790 GOSUB 1480
1800 GOSUB 1590
1810 GOSUB 1710
1820 GOSUB 1840
1830 GOTO 990
```

```
1840 B=7
1850 GOSUB 2200
1860 IF BLOC=2 THEN 3940
1870 GOSUB 3250
1880 RETURN
1890 REM *cas de 9 equations*
1900 GOSUB 1250
1910 GOSUB 1380
1920 GOSUB 1480
1930 GOSUB 1590
1940 GOSUB 1710
1950 GOSUB 1840
1960 GOSUB 1980
1970 GOTO 990
1980 B=8
1990 GOSUB 2200
2000 IF BLOC=2 THEN 3940
2010 GOSUB 3330
2020 RETURN
2030 REM *cas de 10 equations*
2040 GOSUB 1250
2050 GOSUB 1380
2060 GOSUB 1480
2070 GOSUB 1590
2080 GOSUB 1710
2090 GOSUB 1840
2100 GOSUB 1980
2110 GOSUB 2140
2120 GOTO 990
2130 END
2140 B=9
2150 GOSUB 2200
2160 IF BLOC=2 THEN 3940
2170 GOSUB 3430
2180 RETURN
2190 REM* permute 2 equations en cas de p
ivot nul*
2200 IF A(B,B)<>0 THEN 2310
2210 U=U+1
2220 IF B+U<=N THEN 2250
2230 BLOC=2
2240 GOTO 2320
2250 FOR J=1 TO N+1
2260 NA(J)=A(B,J)
2270 A(B,J)=A(U+B,J)
2280 A(U+B,J)=NA(J)
2290 NEXT J
2300 GOTO 2200
2310 BLOC=1
2320 U=0
2330 RETURN
2340 REM *  titre *
2350 PRINT "resolution de": :
2360 PRINT "          systemes lineaires":
 : : : : :
2370 RETURN
2380 REM *attente de la pression d'une to
uche
2390 PRINT "presser une touche pour": :"p
oursuivre"
2400 CALL SOUND(200,1000,0)
2410 CALL KEY(0,K,S)
2420 IF S=0 THEN 2410
2430 RETURN
2440 REM *temporisation*
2450 FOR M=1 TO 200
2460 NEXT M
2470 RETURN
2480 REM *effacement de la ligne 24 de l'
ecran*
2490 CALL HCHAR(24,2,32,28)
2500 RETURN
2510 REM *"quelle" en ligne 24*
2520 CALL HCHAR(24,2,ASC("q"))
2530 CALL HCHAR(24,3,ASC("u"))
2540 CALL HCHAR(24,4,ASC("e"))
2550 CALL HCHAR(24,5,ASC("l"),2)
2560 CALL HCHAR(24,7,ASC("e"))
2570 RETURN
2580 REM*"ligne" en ligne 24*
2590 CALL HCHAR(24,9,ASC("l"))
2600 CALL HCHAR(24,10,ASC("i"))
2610 CALL HCHAR(24,11,ASC("g"))
2620 CALL HCHAR(24,12,ASC("n"))
2630 CALL HCHAR(24,13,ASC("e"))
2640 CALL HCHAR(24,14,63)
2650 RETURN
2660 REM
2670 REM
2680 REM*"colonne" en ligne 24*
2690 CALL HCHAR(24,9,ASC("c"))
2700 CALL HCHAR(24,10,ASC("o"))
2710 CALL HCHAR(24,11,ASC("l"))
2720 CALL HCHAR(24,12,ASC("o"))
2730 CALL HCHAR(24,13,ASC("n"),2)
2740 CALL HCHAR(24,15,ASC("e"))
2750 CALL HCHAR(24,16,63)
2760 RETURN
2770 REM*annule les A(i,j) a partir de i=
2 *
2780 FOR I=2 TO N
2790 FOR J=2 TO N+1
2800 A(I,J)=A(I,J)-(A(I,1)/A(1,1))*A(1,J)
2810 NEXT J
2820 A(I,1)=0
2830 NEXT I
2840 RETURN
2850 FOR I=3 TO N
2860 FOR J=3 TO N+1
2870 A(I,J)=A(I,J)-(A(I,2)/A(2,2))*A(2,J)
2880 NEXT J
2890 A(I,2)=0
2900 NEXT I
2910 RETURN
2920 REM annule les AI3 a partir de I=4
2930 FOR I=4 TO N
2940 FOR J=4 TO N+1
2950 A(I,J)=A(I,J)-(A(I,3)/A(3,3))*A(3,J)
```

```
2960 NEXT J
2970 A(I,3)=0
2980 NEXT I
2990 RETURN
3000 REM  *annule les A(I,4) a partir de
I=5 *
3010 FOR I=5 TO N
3020 FOR J=5 TO N+1
3030 A(I,J)=A(I,J)-(A(I,4)/A(4,4))*A(4,J)
3040 NEXT J
3050 A(I,4)=0
3060 NEXT I
3070 RETURN
3080 REM *annule les A(I,5) a partir de I
=6 *
3090 FOR I=6 TO N
3100 FOR J=6 TO N+1
3110 A(I,J)=A(I,J)-(A(I,5)/A(5,5))*A(5,J)
3120 NEXT J
3130 A(I,5)=0
3140 NEXT I
3150 RETURN
3160 REM *annule les A(I,6( a partir de I
=7*
3170 FOR I=7 TO N
3180 FOR J=7 TO N+1
3190 A(I,J)=A(I,J)-(A(I,6)/A(6,6))*A(6,J)
3200 NEXT J
3210 A(I,6)=0
3220 NEXT I
3230 RETURN
3240 REM *annule les A(I,7) a partir de I
=8*
3250 FOR I=8 TO N
3260 FOR J=8 TO N+1
3270 A(I,J)=A(I,J)-(A(I,7)/A(7,7))*A(7,J)
3280 NEXT J
3290 A(I,7)=0
3300 NEXT I
3310 RETURN
3320 REM *annule les A(I,8) depuis I=9 *
3330 FOR I=9 TO N
3340 FOR J=9 TO N+1
3350 A(I,J)=A(I,J)-(A(I,8)/A(8,8))*A(8,J)
3360 NEXT J
3370 A(I,8)=0
3380 NEXT I
3390 RETURN
3400 REM
3410 REM * annule les A(I,9) depuis I=10
*
3420 REM
3430 FOR I=10 TO N
3440 FOR J=10 TO N+1
3450 A(I,J)=A(I,J)-(A(I,9)/A(9,9))*A(9,J)
3460 NEXT J
3470 A(I,9)=0
3480 NEXT I
3490 RETURN
```

```
3500 REM * transparence
3510 FOR T=1 TO 13
3520 CALL COLOR(T,13,13)
3530 NEXT T
3540 RETURN
3550 REM *retour des caracteres
3560 FOR T=1 TO 13
3570 CALL COLOR(T,12,13)
3580 NEXT T
3590 RETURN
3600 REM* calcul des solutions *
3610 X(N)=A(N,N+1)/A(N,N)
3620 S=0
3630 FOR I=N-1 TO 1 STEP -1
3640 FOR J=I+1 TO N
3650 S=A(I,J)*X(J)+S
3660 NEXT J
3670 X(I)=(A(I,N+1)-S)/A(I,I)
3680 S=0
3690 NEXT I
3700 RETURN
3710 REM * affichage des solutions *
3720 CALL CLEAR
3730 FOR I=1 TO N
3740 PRINT "X";I;"= ";X(I): :
3750 NEXT I
3760 RETURN
3770 REM *commentaires*
3780 PRINT "X1,X2,..XP,..XN sont": :"les
inconnues": : :
3790 PRINT "leur nombre est au plus egal"
: :"a 10": : : : :
3800 RETURN
3810 IF A(N,N+1)<>0 THEN 3850
3820 CALL CLEAR
3830 PRINT "systeme indetermine": : : :
3840 GOTO 1040
3850 CALL CLEAR
3860 PRINT "pas de solutions": : :
3870 GOTO 1040
3880 REM *commentaire sur le type de syst
eme resolu *
3890 CALL CLEAR
3900 PRINT " ce programme traite":"unique
ment des systemes":"carres": : : :
3910 PRINT "c'est a dire qu'il y a       a
utant d'equations que      d'inconnues"
: :
3920 PRINT : : : :
3930 RETURN
3940 CALL CLEAR
3950 PRINT "ce systeme n'est pas carre ,m
eme s'il en donne l'impression": : : :
:
3960 PRINT "il possede soit zero soit   u
ne infinite de solutions": : : :"il n'e
ntre pas dans les  cas traites": : :
3970 GOSUB 2390
3980 GOTO 1030
```

Structure des disques du P-système

Gérard Santraille

Dans le dernier numéro, nous avons passé en revue la plupart des particularités du POS (Pascal Operating System). Nous allons étudier aujourd'hui la structure des disquettes, et voir comment on peut lire et modifier certaines données du système, au moyen des procédures *BLOCKREAD, BLOCKWRITE, UNITREAD et UNITWRITE.*

La première constatation que l'on peut faire est que la sectorisation d'une disquette Pascal est différente de celle du "Disk Manager". En effet, le catalogue d'une disquette Pascal la décrit comme un unique fichier de 358 secteurs et de type "DISPLAY/FIXED 128".
En fait, une telle disquette est constituée par 180 blocs de 512 octets, alors qu'elle est formée de 360 secteurs de 256 octets lorqu'elle est initialisée par le "Disk Manager".

Le DOS (Disk Operating System : Ensemble de routines en Assembleur gérant les entrées/sorties des disques) du TI garde, en Basic, deux secteurs pour le répertoire (512 octets), alors que le POS manipule quatre blocs pour effectuer la même tâche (2 Koctets soit 2048 octets).

Remarque : en Pascal, si l'utilisateur le désire, il peut même demander au système de garder un double du répertoire (en répondant "Yes" à la question "Duplicate Dir ?" de la commande Zéro du Filer). Le POS remet alors à jour les deux répertoires (l'original et la copie) chaque fois qu'il manipule un fichier (notons au passage que toutes les versions du système UCSD n'offrent pas cette intéressante particularité). Une disquette comprenant un double catalogue perd bien évidemment 2048 octets de capacité de stockage.

Avant de poursuivre, voyons rapidement comment utiliser un double répertoire.
Lorsqu'on écrit directement des données sur un disque, au moyen des procédures *UNITWRITE* ou *BLOCKWRITE*, il peut arriver que l'on écrase le répertoire de la disquette (qui se situe physiquement sur les blocs 2 à 5), cette mésaventure est relativement fréquente dès que l'on manipule "illégalement" certains blocs. Une copie du répertoire (qui, elle, se situe sur les blocs 6 à 9) est alors précieuse puisqu'il suffit d'exécuter l'utilitaire *COPYDUPDIR* pour rétablir la situation.

Listing 1

```
Type date      = packed record
                    mois : 1..12;
                    jour : 1..31;
                    annee: 0..99
                 end;

     fichtype = (inconnu,mauvais,code,texte,info,
                 donnees,graphique,photo);

     diskinfo = record
                    ext1       : integer;
                    bloc1      : integer;
                    disktype   : integer;
                    nom        : string[7];
                    blocs      : integer;
                    nbrefich   : integer;
                    ext2       : integer;
                    date_sys   : date;
                    ext3,ext4: integer
                 end;

     fichier  = record
                    blocdeb    : integer;
                    prebloc    : integer;
                    categorie: fichtype;
                    nom        : string[15];
                    octlibre   : integer;
                    dermodif   : date
                 end;

     catalogue= record
                    volume     : diskinfo;
                    rep        : array[1.77] of fichier
                 end;
```

Listing 2

```
PROGRAM MODIFDATE;

TYPE DATE=PACKED RECORD
             MOIS :1..12;
             JOUR :1..31;
             ANNEE:0..99
           END;

     OCTET=0..255;

     REPERTOIRE=RECORD
        TAMPON1:PACKED ARRAY[1..20] OF OCTET;
        DATESYS:DATE;
        TAMPON2:PACKED ARRAY[1..490] OF OCTET
      END;

VAR  REP:REPERTOIRE;
     J:1..31;

PROCEDURE AFFICHEDATE(DD:DATE);
```

Analysons le contenu d'une disquette.
Les deux premiers blocs (0 et 1) contiennent un programme de lancement (en langage machine) et des tables d'initialisation. Cette zone est cachée à l'utilisateur (qui ne s'en soucie d'ailleurs pas), et n'apparaît pas lors de l'affichage du répertoire des fichiers. Ce programme est écrit durant le formattage de la disquette (utilitaire *DFORMAT*) et est, par la suite, entièrement manipulé par le système.

Les quatre blocs suivants contiennent donc le répertoire de la disquette, qui peut être également dupliqué dans les quatre autres suivants.
Une disquette à répertoire non dupliqué offre donc 180 − 2 − 4 = 174 blocs à l'utilisateur (89088 octets).

Analysons maintenant la structure du répertoire :

Voir listing 1

Remarques :

- Si l'on évalue la taille du type catalogue (au moyen de la fonction SIZEOF), on s'aperçoit qu'elle est de 2028 octets, et non pas de 2048 (4 fois 256); il reste en effet 20 octets inutilisés.
- En mettant en parallèle les déclarations des enregistrements "diskinfo" et "fichier", on constate qu'ils ont non seulement des champs de même nature, mais également un encombrement mémoire identique (26 octets). Le type catalogue se présente alors comme 78 enregistrements de 26 octets; le premier contenant des informations sur le volume et les 77 suivants sur chaque fichier (on notera au passage que 77 est le nombre maximal de fichiers présents sur une même disquette).

Voyons plus en détail la signification et le rôle de chaque terme.

L'enregistrement "date" se passe de commentaire. Notons simplement qu'il s'agit d'une variable compactée et qu'elle n'occupe ainsi que 2 octets.
"Fichtype" indique le type du fichier; il s'agit d'un ensemble explicite dont on reconnaît la plupart des éléments.
"Inconnu" ne se rencontre jamais car le système détruit automatiquement tout fichier non reconnu.
"Mauvais" marque les blocs défectueux du disque (repérés par la commande *Bad Blocks* du Filer).
"Texte Code" et "Données" sont bien connus.
"Info" correspondra à des fichiers temporaires du Debugger, quand il sera disponible.
Enfin, "Graphique" et "Photo" sont également des extensions futures.

```
BEGIN
  WITH DD DO
    BEGIN
    WRITE(JOUR:2);
    WRITE(' ');
    CASE MOIS OF
      1:WRITE('JANVIER');
      2:WRITE('FEVRIER');
      3:WRITE('MARS');
      4:WRITE('AVRIL');
      5:WRITE('MAI');
      6:WRITE('JUIN');
      7:WRITE('JUILLET');
      8:WRITE('AOUT');
      9:WRITE('SEPTEMBRE');
     10:WRITE('OCTOBRE');
     11:WRITE('NOVEMBRE');
     12:WRITE('DECEMBRE')
    END;
    WRITELN(' 19',ANNEE:2)
  END;
END;


BEGIN
UNITREAD(4,REP,512,2);
WITH REP,DATESYS DO
  BEGIN
  WRITE('LA DATE SYSTEME EST LE ');
  AFFICHEDATE(DATESYS);
  WRITE('JOUR (1..31) ? ',CHR(7));
  READLN(J);
  JOUR:=J;
  PAGE(OUTPUT);
  AFFICHEDATE(DATESYS)
  END;
UNITWRITE(4,REP,512,2)
END.
```

Listing 3

```
Program DIR;

type date      = packed record
                    mois : 1..12;
                    jour : 1..31;
                    annee: 0..99
                 end;

     fichtype = (inconnu,mauvais,code,texte,info,
                 donnees,graphique,photo);

     diskinfo = record
                    ext1      : integer;
                    bloc1     : integer;
                    disktype  : integer;
                    nom       : string[7];
                    blocs     : integer;
                    nbrefich  : integer;
                    ext2      : integer;
                    date_sys  : date;
                    ext3,ext4: integer
                 end;

     fichier  = record
                    blocdeb   : integer;
                    prebloc   : integer;
```

L'enregistrement "diskinfo" contient, comme son nom l'indique, diverses informations propres à la disquette.
"Ext1", "Ext2", "Ext3" et "Ext4" sont des extensions futures; ces quatre entiers ne sont donc pas manipulés par le système.
"Bloc1" indique quel est le premier bloc utilisable du disque (6 ou 10 selon que le répertoire est dupliqué ou non).
"disktype" est un entier toujours nul sensé déterminer dans le futur le type du disque (??).
"Nom" est bien entendu le nom du disque.
"blocs" est le nombre de blocs de ce dernier (180).
"nbrefich" est le nombre de fichiers présents et "date-sys" la date du système.

Nous verrons par la suite un petit programme permettant de modifier la date sans passer par le Filer.

Voyons maintenant la structure de fichier. "Blocdeb" est le numéro du bloc où commence le fichier, alors que "prebloc" indique le numéro du premier bloc suivant ce dernier. La longueur d'un fichier vaut donc "prebloc" – "blocdeb".
"Categorie" indique le type de fichier, (pas son nom), "dermodif" la date de la dernière mise à jour et "octlibre" le nombre d'octets non utilisés dans le dernier bloc du fichier (uniquement pour les fichiers de données).

Examinons enfin la structure du catalogue. Elle se compose de 78 enregistrements; le premier de type "diskinfo" et les 77 suivants de type "fichier".

Programmes d'application

Modification de la date sans passer par le Filer

Le programme que je vous propose lit le premier bloc du répertoire (on ne s'intéresse qu'à la date; il est donc inutile de le lire en entier), modifie la date (uniquement le jour) et écrit à nouveau ce bloc sur la disquette.

Voir listing 2

L'analyse du listing suscite plusieurs remarques :

1) comme on ne s'intéresse qu'à la date, la structure complète du répertoire n'a pas été explicitée : on s'est contenté de placer deux tampons;
2) la deuxième remarque complète la première : le deuxième tampon est obligatoire car la plus petite quantité que le contrôleur sache manipuler, au moyen de *UNITWRITE*, est le bloc. Si l'on avait seulement lu et

```
                categorie: fichtype;
                nom       : string[15];
                octlibre  : integer;
                dermodif  : date
              end;

      catalogue= record
                volume    : diskinfo;
                rep       : array[1..77] of fichier
              end;

var cat   : catalogue;
    nodisk,numerr,i: integer;
    car:char;

procedure ioerreur(num:integer);

begin
writeln('Erreur de lecture');
  case num of
    1: write('block endommage, parite erronee');
    2: write('mauvais numero de lecteur');
    3: write('operation illegale');
    4: write('lecture stoppee par l''utilisateur');
    5: write('lecteur non reconnu');
    6: write('fichier inconnu');
    7: write('nom incorrect');
    8: write('espace memoire insuffisant');
    9: write('volume inconnu');
   10: write('fichier inconnu');
   11: write('homonymie de fichiers');
   12: write('fichier deja ouvert');
   13: write('fichier ferme');
   14: write('format de lecture incorrect');
   15: write('debordement de tampon');
   16: write('volume protege contre l''ecriture');
   17: write('numero de block illegal');
   18: write('tampon illegal')
  end;
writeln(chr(7))
end;

procedure aff_fichier( fich:fichier);

 procedure aff_date(dd:date);

 begin
  with dd do
   begin
   write(jour:3,'-');
   case mois of
     1:write('Jan');
     2:write('Fev');
     3:write('Mar');
     4:write('Avr');
     5:write('Mai');
     6:write('Jun');
     7:write('Jul');
     8:write('Aou');
     9:write('Sep');
    10:write('Oct');
    11:write('Nov');
    12:write('Dec')
   end;
   write('-',annee:2,' ')
  end
 end;

 procedure aff_type(genre:fichtype);

 begin
  case genre of
    inconnu   : write('inconnu');
    mauvais   : write('mauvais');
    code      : write('code');
    texte     : write('texte');
    info      : write('info');
    donnees   : write('donnees');
    graphique : write('graphique');
    photo     : write('photo')
  end;
 write(' ')
 end;
```

écrit 22 octets (premier tampon et date) selon :

UNITREAD(4,REP,22,2);
UNITWRITE(4,REP,22,2);

on aurait écrasé 490 octets du premier bloc du répertoire (je le sais par expérience) et perdu l'accés à tous les fichiers.

Rappelons à ce sujet l'utilisation des procédures *UNITREAD, BLOCKREAD, UNITWRITE, BLOCKWRITE.*

⋆UNITWRITE / UNITREAD

4 paramètres dans son utilisation classique :
UNITREAD(n,var,long,dep) avec :

n : numéro du périphérique (4, 5 ou 9 pour un lecteur de disquettes);
var : variable à lire;
long : nombre d'octets à lire (long doit être égal à la taille de var);
dep : numéro du bloc de départ si le périphérique est structuré.

L'utilisation de *UNITWRITE* est en tout point identique si ce n'est que pour un lecteur de disquettes, *long* doit être un multiple de 512.

⋆BLOCKREAD / BLOCKWRITE

Il s'agit d'une fonction qui retourne le nombre de blocs effectivement transférés :

n:=BLOCKREAD(fich,var,nbloc,dep) avec :

n : nombre de blocs transférés;
fich : nom du fichier d'où se fait le transfert (doit avoir été préalablement activé par l'instruction RESET (fich,'...');
var : variable lue;
nbloc : nombre de blocs à transférer;
dep : bloc de départ.

L'emploi de *BLOCKREAD* est identique en tout point.

Si, dans le programme "date", il était possible de remplacer *UNITREAD* par *BLOCKREAD* il est par contre impossible de substituer *BLOCKWRITE* à *UNITWRITE.* En effet l'emploi de *BLOCKWRITE* suppose que l'on ait ouvert le fichier par un "RESET" et qu'on le ferme au moyen de CLOSE(fich,lock). Or,

```
begin {aff fichier}
  with fich do
  begin
  write(nom,'':15-length(nom),prebloc-blocdeb:3);
  aff_date(dermodif);
  aff_type(categorie);
  writeln;
  end;
end;


{PROGRAMME PRINCIPAL}

begin
page(output);
writeln('CATALOGUE D''UNE DISQUETTE');
write('Numero du lecteur (4,5 ou 9) : ',chr(7));
readln(nodisk);
writeln;
unitread(nodisk,cat,2028,2);
numerr:=ioresult;
if numerr<>0 then ioerreur(numerr)
             else
  begin
    with cat do
    begin
     writeln(volume.nom,':');
     for i:=1 to volume.nbrefich do
      begin
      aff_fichier(rep[i]);
      if i mod 19=0 then
        begin
        write('Tapez <espace> pour poursuivre',chr(7));
        read(keyboard,car);
        page(output);
        writeln;
        writeln
        end
      end
    end
  end
end.
```

chaque fermeture d'un fichier qui a été modifié provoque la recopie du répertoire, de la mémoire de l'ordinateur vers la disquette concernée. Le répertoire que l'on a soigneusement modifié se voit alors écrasé et irréméédiablement perdu. L'emploi de *BLOCKWRITE* est donc proscrit quand il s'agit de modifier un répertoire.

Catalogue d'une disquette sans passer par le Filer

Ce programme vous permet d'afficher à l'écran le contenu d'une disquette; à savoir le nom de chaque fichier, son type, sa date de création, sa taille. La présentation est du type de celle faite par le Filer. A vous de modifier ce programme pour qu'il puisse afficher le catalogue sur l'imprimante ou indiquer la positon physique de chaque fichier.

Voir listing 3

La description des divers enregistrements du catalogue est ici complète. Le programme saisit le numéro du lecteur, (un traitement des erreurs d'entrées/sorties est assuré par la fonction *IORESULT*) et affiche la liste des fichiers du volume concerné.

La connaissance de la structure du catalogue permet ainsi de créer ses propres utilitaires, les applications ne manquent pas, un dernier conseil : dupliquez les disquettes ainsi manipulées ou dupliquez le répertoire... ■

Poèmes et graphismes

Philippe Misandeau

J'étais assis là depuis déjà bien longtemps, sur le bureau, entre le cendrier en albâtre où finissait de se consumer la 17ème cigarette du paquet cartonné, et ces drôles de cassettes qui n'entendent décidément pas grand-chose à la musique. Oh, je n'avais pas encore eu le temps de m'ennuyer...

Lui, je le connaissais bien. En quelques cycles, j'avais beaucoup appris sur ses capacités mentales : besogneux, attentif, méticuleux voire maniaque, artiste à ses heures, assez patient; j'avais mis à rude épreuve sa logique, l'avais obligé à délier ses doigts, comme à tanner le cuir de ses fesses sur la chaise inconfortable de notre table de travail !

Elle, je ne l'avais que peu rencontrée jusqu'à aujourd'hui. Bien sûr, nous nous entrevîmes au magasin et c'est elle qui m'avait introduit ici, le jour

de son anniversaire à lui. Je dois dire que, même lors de ces brefs moments, son regard bleu, l'ondulation blonde de ses longs cheveux et le timbre de sa voix (surtout la mélodie enchanteresse de sa voix), avaient produit beaucoup d'effet sur moi : sa douce image restait bien ancrée au plus profond de ma mémoire.

Je ne vais pas m'étendre sur nos premiers contacts avec lui... Voyez cependant une de nos premières réalisations. Ce programme graphique simple m'a demandé beaucoup, beaucoup de patience. A coups de "messages d'erreur", de "beeps" et de silence, j'ai pu le mettre sur la voie de ce travail; je vous raconte cela pour la petite histoire, mais il m'a même fallu parfois intervenir directement. J'ai, par exemple, inventé un *"PRESS ALPHA LOCK PLEASE !"*. Oh, ne cherchez bien sûr pas ce message d'erreur sur votre TI préféré, il n'existe pas ! Cela faisait au moins une heure qu'en vain, il cherchait à charger un programme avec un "old cs1" en minuscules ! Mon purisme m'interdisait d'accepter une telle commande et puis, jouer un peu avec ses nerfs justifie mon amour pour la grammaire ! Il a été assez surpris de lire ce message sur son écran TV. Peut-être à cause du *"PLEASE !"* emprunt d'une certaine condescendance qui m'avait échappé. Il a dodeliné du chef, m'a regardé de biais, l'oeil aussi plissé qu'était mal rasée sa lèvre supérieure, immobilisée d'abórd sous l'emprise d'une moue fort plaisante. Et toc, son index gauche a débloqué la situation. Le lévrier afghan, choqué, préféra changer de tapis lorsqu'un "OUAIS" retentissant lui appris que son maître devenait de plus en plus exubérant et, par voie de conséquence, de moins en moins fréquentable... Personne n'avait vraiment pris conscience de mon intervention. En tous cas, nous étions momentanément relancés... et moi, libéré de la frappe inutile de ses gros doitgs sur mon clavier.

Je suis assez fier du dernier logiciel graphique que nous avons réalisé ensemble. S'il est courageux, ce sera le point de départ d'un grand fichier d'auteurs réalisateurs... Pour sortir cela, j'ai fait le plein de presque toutes mes mémoires vives. Mais regardez... il ne nous manque "plus que la parole", non ? C'est un peu à cause de cela d'ailleurs que tout est arrivé. Il a voulu y mettre de la musique, et c'est ELLE la musicienne à la maison ! Sa tessiture est... différente de la mienne, mais j'en reconnais la beaûté, la clarté, la pureté. Quand elle chante, tout se bouleverse autour d'elle. Je l'ai déjà admirée lorsqu'elle s'attarde dans la salle de bain. Si les portes sont ouvertes, l'on entend bien le galbe et les rondeurs de la mélodie. Est-ce à cause de moi si, dans ces moments, les programmes sur lequels il s'acharnait perdaient des lignes et ne tournaient plus ?

Il voulait un nouveau Purcell pour accompagner ses talents graphiques. Alors elle est venue embaumer la pièce. Tout devint lumière. Les fleurs de la tapisserie frémirent sur son passage. Sa robe légère a caressé mon écran - *CALL SCREEN(7)* - et ses doigts de fée ont effleuré mes touches : "Que je renaisse et reconnaisse un monde habité par le chant". Même Aragon y aurait perdu son ASCII. Qu'est-ce que cela a donné avant qu'il ne me débranche ? J'en étais à : *AGAIN, AGAIN, AGAIN...* ■

```
100 REM ********************
110 REM *     Poemes de    *
120 REM *  Raymond Queneau *
130 REM ********************
140 REM *      Basic TI    *
150 REM ********************
160 REM
170 CALL CLEAR
180 CALL SCREEN(5)
190 DATA Ce programme peut creer cent,mil
le milliards de sonnets re-,guliers. Je
 peux vous les fai-
200 DATA re tous et en lisant 24h/24,vous
 aurez de la lecture pour,190 258 751 a
nnees plus quel-
210 DATA ques plombes et broquilles ce-,l
a en lisant toute la journee,et 365 jou
rs par ans... (mais
220 DATA si l'on compte 50 secondes,pour
lire un sonnet et 5 se-,condes pour tou
rner la page a
230 DATA 8 heures par jours et 200 jours,
par an vous n'aurez que pour,un million
 de siecles de lec-
240 DATA ture !!). Je vous laisse,commenc
er... A moins que vous,ne vouliez refai
re le calcul...
250 DATA Chaque sonnet a un theme et,une
continuite pour la lisibi-,lite et le c
harme - a defaut
260 DATA d'etre parfaitement translucide,
" ",Appuyez sur <ENTER>...
270 DATA 1,3,5,7,11,13,15,17,11,13,15,9,1
1,13
280 DATA Le roi de la pampa retourne sa`c
hemise,Le cheval parthenon s'enerve`sur
 sa frise
290 DATA Le vieux marin breton de tabac`p
rit sa prise,C'etait a cinq o'clock qu'
il`sortait la marquise
300 DATA Du jeune avantageux la nymphe`et
ait eprise,Il se penche il voudrait`att
raper sa valise
310 DATA Quand l'un avecque l'autre`aussi
tot sympatise,Lorsqu'un jour exalte l'a
ede`prosaise
320 DATA Le marbre pour l'acide est une`f
riandise,Lorsque tout est fini lorsque`
l'on agonise
330 DATA Pour la mettre a secher aux`corn
es des taureaux,Depuis que lord elgin n
egligea`ses naseaux
340 DATA Pour du fin fond du nez`exciter
les arceaux,Pour consommer un the puis
des`petits gateaux
350 DATA Snob un peu sur les bords des`bo
rds fondamentaux,Que convoitait c'est s
ur une`horde d'escrocs
360 DATA Se faire il pourrait bien que`ce
 soit des jumeaux,Pour deplaire au prof
ane aussi`bien qu'aux idiots
370 DATA D'aucuns par dessus tout`prisent
 les escargots,Lorsque le marbrier asti
que`nos tombeaux
380 DATA Le cornedebif en boite empeste`l
a remise,Le turc de ce temps-la`patauge
ait dans sa crise
```

```
390 DATA Sur l'antique bahut il choisit`s
a cerise,Le chauffeur indigene`attendai
t dans la brise
400 DATA Une toge il portait qui`n'etait
pas de mise,Il se penche et alors a sa`
grande surprise
410 DATA La decouverte alors voila qui`tr
aumatise,La critique lucide apercoit ce
`qu'il vise
420 DATA Sur la plage un forain de feu`se
 gargarise,Des etres indecis vous parle
nt`sans franchise
430 DATA Et fermentent de meme et les`cui
rs et les peaux,Il chantait tout de mem
e oui`mais il chantait faux
440 DATA Il n'avait droit qu'a une et`le
jour des rameaux,Elle soufflait bien fo
rt par`dessus les coteaux
450 DATA Des narcisses on cueille ou`bien
 on est des veaux,Il ne trouve aussi se
c qu'un`sac de vieux fayots
460 DATA On espere toujours etre de`vrais
 normaux,Il donne a la tribu des cris`a
ux sens nouveaux
470 DATA Qui sait si le requin boulotte`l
es turbots ?,Et tout vient signifier la
 fin`des haricots
480 DATA Je me souviens encor de cette`he
ure exeuquise,Le cheval parthenon`friss
onnait sous la bise
490 DATA Souvenez-vous amis de ces iles`d
e frise,On etait bien surpris par`cette
 plaine grise
500 DATA Quand on prend des photos de`cet
te tour de pise,Il deplore il deplore u
ne`telle mainmise
510 DATA Et pourtant c'etait lui le`frere
 de feintise,L'un et l'autre a raison n
on`la foule insoumise
520 DATA Du voisin le papou sucotte`l'apo
physe,On nous fait devenir une orde`mar
chandise
530 DATA Les gauchos dans la plaine`agita
ient leurs drapeaux,Du climat londonien
 ou`s'ebattent les beaux
540 DATA Ou venaient par milliers`s'echou
er les harenceaux,Quand se carbonisait
la fureur`des chateaux
550 DATA D'ou galilee jadis jeta ses`peti
ts pots,Qui se plait a flouer de`pauvre
s provinciaux
560 DATA Qui clochard devenant jetait`ses
 oripeaux,Le vulgaire s'entete a vouloi
r`des vers beaux
570 DATA Que n'a pas devore la horde`des
mulots ?,On prepare la route aux`penser
s sepulcraux
580 DATA Nous avions aussi froid que`nus
sur la banquise,Il grelottait le pauvre
 aux`bords de la tamise
590 DATA Nous regrettions un peu ce tas`d
e marchandises,Un audacieux baron empoc
he`toute accise
600 DATA D'une etrusque inscription la`pi
erre etait incise,Aller a la grand vill
e est`bien une entreprise
610 DATA Un frere meme bas est la part`in
decise,L'un et l'autre ont raison non`l
a foule imprecise
620 DATA Le gourmet en salade avale la`cy
tise,De la mort on vous greffe une`orde
 batardise
630 DATA Lorsque pour vous distraire y`pl
antions nos treteaux,Quand les grelons
fin mars`mitraillent les bateaux
640 DATA Lorsqu'on voyait au loin`flamber
 les arbrisseaux,Lorsque vient le pompi
er avec`ses grandes eaux
650 DATA Les grecs et les romains en`vain
 cherchent leurs mots,Elle effraie le b
erry comme`les morvandiaux
660 DATA Que les parents feconds`offrent
aux purs berceaux,A tous n'est pas donn
e d'aimer`les chocs verbaux
670 DATA L'enfant pur aux yeux bleus`aime
 les berlingots,La mite a grignote tiss
us os`et rideaux
680 DATA Du pole a rosario fait une`belle
 trotte,La grece de platon a coup sur`n
'est point sotte
690 DATA On seche le poisson dorade ou`mo
lve lotte,Du gange au malabar le lord`a
nglais zozotte
700 DATA L'esprit souffle et resouffle`au
-dessus de la botte,Devant la boue urba
ine on`retrousse sa cotte
710 DATA La genealogiste observe leur`bou
illote,Le poete inspire n'est point`un
polyglotte
720 DATA Le loup est amateur de coq et`de
 cocotte,Le brave a beau crier ah`cre-n
om saperlotte
730 DATA Aventures on eut qui s'y pique`s
'y frotte,On comptait les esprits acere
s`a la hotte
740 DATA On sale le requin on fume a`l'ec
halotte,Comme a chandernagor le manant`
sent la crotte
750 DATA Le touriste a florence ignoble`c
haribotte,On gifle le marmot qui plonge
`sa menotte
760 DATA Gratter le parchemin deviendra`s
a marotte,Une langue suffit pour emplir
`sa cagnotte
770 DATA Le chat fait un festin de`tetes
de linottes,Le lache peut arguer de sa`
mine palotte
780 DATA Lorsque socrate mort passait`pou
r un lutin,Lorsqu'on boit du mate l'on`
devient argentin
```

```
790 DATA Lorsqu'on revient au port en`ess
uyant un grain,Le colonel s'eponge un b
lason`dans la main
800 DATA L'autocar ecrabouille un peu`d'e
sprit latin,Lorsqu'il voit la gadoue il
`cherche le purin
810 DATA Il voudra retrouver le germe`adu
lterin,Meme s'il prend son sel au`celte
 c'est son bien
820 DATA Le chemin vicinal se nourrit`de
crottin,Les croque-morts sont la pour`s
e mettre au turbin
830 DATA L'amerique du sud seduit les`equ
ivoques,Sa sculture est illustre et`dan
s le fond des coques
840 DATA Enfin on vend le tout homards`et
 salicoques,Ne fallait pas si loin agit
er`ses breloques
850 DATA Les rapports transalpins`sont-il
s biuniviques ?,On regrette a la fin le
s`agrestes bicoques
860 DATA Frere je te comprends si`parfois
 tu debloques,Barde que tu me plais tou
jours`tu soliloques
870 DATA On a bu du pinard a toutes les`e
poques,Cela considerant o lecteur tu`su
ffoques
880 DATA Exaltent l'espagnol les`oreilles
 baroques,On transporte et le marbre et
`debris et defroques
890 DATA On s'excuse il n'y a ni`baleines
 ni phoques,Les indes ont assez sans ca
 de`pendeloques
900 DATA Les banquiers d'avignon`changent
-ils les baioques ?,On mettait sans fac
on ses plus`infestes loques
910 DATA Frere je t'absoudrai si tu`m'emb
erlucoques,Tu me stupefies plus que tou
s`les ventriloques
920 DATA Grignoter des bretzels`distrait
bien des colloques,Comptant tes abattis
 lecteur`tu te disloques
930 DATA Si la cloche se tait et son`terl
intintin,Si l'europe le veut l'europe`o
u son destin
940 DATA Le mammifere est roi nous`sommes
 son cousin,L'ecu de vair ou d'or ne du
re`qu'un matin
950 DATA Le beaune et le chianti`sont-ils
 le meme vin ?,Mais on n'aurait pas vu
le`metropolitain
960 DATA La gemellite vraie accuse son`de
stin,Le metromane a force incarne`le de
vin
970 DATA Mais rien ne vaut grille le`morc
eau de boudin,Toute chose pourtant doit
`avoir une fin
980 OPTION BASE 1
990 DIM LP(14),L1$(10),LR$(14)
1000 FOR B=1 TO 12
1010 CALL COLOR(B,8,1)
1020 NEXT B
1030 FOR B=1 TO 24
1040 READ T$
1050 GOSUB 1820
1060 NEXT B
1070 FOR B=1 TO 14
1080 READ LP(B)
1090 NEXT B
1100 CALL KEY(0,K,S)
1110 IF K<>13 THEN 1100
1120 CALL CLEAR
1130 CALL SCREEN(13)
1140 FOR I=1 TO 14
1150 FOR B=1 TO 10
1160 READ L1$(B)
1170 NEXT B
1180 RANDOMIZE
1190 T$=L1$(INT(RND*10)+1)
1200 LR$(I)=T$
1210 GOSUB 1640
1220 IF LP(I)<15 THEN 1320
1230 IF (LP(I)=15)*(I=11)THEN 1290
1240 IF LP(I)<>17 THEN 1320
1250 FOR D=0 TO 50
1260 NEXT D
1270 PRINT : : : : : : : : : :
1280 GOTO 1320
1290 FOR D=0 TO 50
1300 NEXT D
1310 PRINT : : : : : : : : : :
1320 NEXT I
1330 T$="<A> pour arret"
1340 B=22
1350 GOSUB 1820
1360 T$="<M> pour meme poeme"
1370 B=23
1380 GOSUB 1820
1390 T$="<N> pour nouveau poeme"
1400 B=24
1410 GOSUB 1820
1420 CALL KEY(0,K,S)
1430 IF K=65 THEN 1880
1440 IF K=77 THEN 1480
1450 IF K<>78 THEN 1420
1460 RESTORE 280
1470 GOTO 1120
1480 CALL CLEAR
1490 FOR I=1 TO 14
1500 T$=LR$(I)
1510 GOSUB 1640
1520 IF LP(I)<15 THEN 1620
1530 IF (LP(I)=15)*(I=11)THEN 1590
1540 IF LP(I)<>17 THEN 1620
1550 FOR D=0 TO 50
1560 NEXT D
1570 PRINT : : : : : : : : : :
1580 GOTO 1620
1590 FOR D=0 TO 50
1600 NEXT D
```

```
1610 PRINT : : : : : : : : : : :
1620 NEXT I
1630 GOTO 1330
1640 PC=1
1650 L=LP(I)
1660 C=2
1670 LO=LEN(T$)-1
1680 CR=0
1690 CAR=ASC(SEG$(T$,PC,1))
1700 IF CAR<>96 THEN 1750
1710 C=2
1720 L=L+1
1730 PC=PC+1
1740 GOTO 1690
1750 CALL HCHAR(L,C,CAR)
1760 CR=CR+1
1770 IF CR<LO THEN 1790
1780 RETURN
1790 C=C+1
1800 PC=PC+1
1810 GOTO 1690
1820 PC=1
1830 FOR X=2 TO 1+LEN(T$)
1840 CALL HCHAR(B,X,ASC(SEG$(T$,PC,1)))
1850 PC=PC+1
1860 NEXT X
1870 RETURN
1880 CALL CLEAR
1890 END
```

```
100 REM ********************
110 REM *     Programme      *
120 REM *     graphique      *
130 REM ********************
140 REM *     Basic  TI      *
150 REM ********************
160 REM
170 CALL CLEAR
180 DATA 03,0000000000000103
190 DATA FFFFFFFF0F0C0C,C08,000000000001
070F,00000000000000FF,F0FFFFFFFFFFFFFF
200 DATA 21FFFFFFFFFFFFFF,0000C0F0F0FCFFF
F,0F317C7CFFFFFFFF,00000000E0F0FFFF
210 DATA 0303070707070707,FEFFFFFFFFFFFFF
F,7F3F9F5FEFDFCF9C,0F0F0F0F1F1F1F1F
220 DATA FFFFFFFFFFFFFEFE,F0C0C0808,1F070
3,FFFFDC00000000C7,0F0F0103030707FF
230 DATA 7F7F7F7F7F7F7F7F,0F1F1F3F3F3F7F7
F,FCFCFCFCF8F8F8F8,172800000000000F
240 DATA FF2000000000077F,FFFF7F3F070301E
1,F0F0C0C0000000F,F8F0F8FCFDFFFFFF
250 DATA 0F3F7FFFFF7F3F1F,80C0E1FFFFF7C18
1,7FFFFFFFFFFFFFFF,7F3F3F3F7F7F7F7F
260 DATA F7E3C199B9A0A0C8,FFFEF8F8F0F8F0F
,FFFC6,C9C9C5E5E1F1F1F9,0000000080C8FCF
8
270 DATA 0000000000001013,FFFF7F3F1F0F1FF
F,C0C0F0F0C080008
280 DATA 3F1F1F0F0F070707,FFFFFFFEF8E0C08
,0F070508081B3FF,FFFF4F0F3FFFFF7F
290 DATA C0E0F0F0F0F8FCFE,030303030101010
1,0000000080E0E,840E0E1FFF7F1F0F
300 DATA 0F1F3FFFFFFFFFFF,FFFF7F7F7F3F0F0
7,0000000003CFFFFF,6F1F7FFFFFFFFFFF
310 DATA 1F3F7F0F06,FFFFFFFF7F3F1F1F,FFFF
FFFFFFFCF0FC,1F1F3F7F7F7F7F7F
320 DATA 3F3F3F1F0701,FFFFFFFFFFFFFFFF,00
030F1F7FFFFFFF,FFFFFFFF3FC7C3F9
330 DATA F9FFFFFFFFFFFFFF
340 DATA 32,32,32,32,32,101,101,32,32,32,
32,32,32
350 DATA 32,97,154,103,153,153,153,153,10
2,104,101,32,32
360 DATA 97,105,155,153,153,153,153,153,1
53,153,102,106,32
370 DATA 107,153,108,109,153,153,153,153,
153,153,153,108,142
380 DATA 110,153,111,112,113,114,97,115,1
53,153,153,118,32
390 DATA 117,153,118,97,119,32,120,121,15
3,153,153,122,32
400 DATA 116,153,123,124,153,125,126,153,
153,153,153,135,32
410 DATA 127,128,153,96,130,96,153,153,15
3,153,129,99,32
420 DATA 127,131,153,132,32,133,134,153,1
53,153,135,32,32
430 DATA 136,156,137,99,32,138,139,153,15
3,153,140,32,32
440 DATA 141,153,142,32,96,143,144,153,15
3,153,142,32,32
450 DATA 96,145,32,32,100,146,147,153,153
,153,98,32,32
460 DATA 32,96,99,32,148,149,153,153,153,
150,99,32,32
470 DATA 32,32,32,32,32,151,153,153,98,99
,32,32,32
480 DATA 32,32,32,32,32,152,153,98,32,32,
32,32,32
490 CALL SCREEN(8)
500 FOR B=1 TO 16
510 CALL COLOR(B,5,1)
520 NEXT B
530 PRINT TAB(13);"JE": :TAB(11);"CHERCHE
": :TAB(13);"MON": :TAB(11);"CRAYON": :
 : : : : : : : : :
540 FOR B=96 TO 156
550 READ C$
560 CALL CHAR(B,C$)
570 NEXT B
580 FOR L=6 TO 20
590 FOR C=10 TO 22
600 READ V
610 CALL HCHAR(L,C,V)
620 NEXT C
630 NEXT L
640 CALL KEY(0,K,S)
650 IF S=0 THEN 640
660 CALL CLEAR
670 END
```

Un désassembleur en assembleur

Jean-Luc Bazanegue

Cet utilitaire transforme le code machine, situé en mémoire vive ou morte, en un fichier source compatible avec l'Editeur/Assembleur Texas Instruments. Il est donc possible, si le produit du désassemblage est dirigé vers une disquette, de traiter le fichier ainsi créé exactement de la même façon qu'un fichier source classique.

Le programme fonctionne indifféremment avec l'Editeur/Assembleur et la Mini-mémoire (une extension de mémoire est indispensable).

Ce programme est relativement complexe et une analyse complète de son fonctionnement occuperait la totalité de ce numéro. Nous avons donc décidé de limiter cet article à un "mode d'emploi" du logiciel.

Adresses à désassembler

La première question posée par le programme, lorsque vous lancez son exécution, concerne la zone de mémoire à désassembler. Vous devez entrer successivement la première et la dernière adresse, en hexadécimal. Si vous faites des fautes de frappe, les corrections sont possibles grâce aux touches *FCTN-S* (flèche vers la gauche) et *FCTN-D* (flèche vers la droite).

Certaines erreurs peuvent provoquer la génération d'un fichier source incohérent. Pour éviter cela, la saisie des adresses est protégée de la manière suivante :

- le programme n'accepte que les caractères propres à l'écriture d'un nombre en hexadécimal (0 à 9 et A à F). L'entrée d'un autre caractère provoque la remise à zéro du champ et le curseur se place en première position;
- les adresses impaires sont rejetées systématiquement (toutes les instructions du TMS 9900 utilisent au moins un mot);
- le nombre représentant la dernière adresse à désassembler doit être logiquement supérieur au nombre représentant la première adresse. Sinon, le programme retourne à la première question. Cette particularité peut être utilisée en cas d'erreur : si, par exemple, vous avez entré A000 comme adresse de départ alors que la zone à désassembler débute en A100, communiquez au programme une valeur inférieure à A000, les deux adresses seront alors ignorées et vous pourrez frapper le nombre désiré.

La touche *"BACK"* (FCTN-9) autorise le retour à l'Editeur/Assembleur (ou à la mini-mémoire).

Note : la dernière adresse désassemblée ne sera pas forcément celle prévue. En effet, suivant les modes d'adressage utilisés, une seule instruction peut occuper jusqu'à 3 mots (6 octets). Exemple :

- vous communiquez au programme B000 comme dernière adresse;
- cette adresse contient la valeur C9E0 qui correspond, en code mnémonique, à MOV@>xxxx,@>yyyy(R7);
- dans ce cas, les adresses B002 et B004 contiennent respectivement les adresses >xxxx et >yyyy et la dernière adresse désassemblée sera B004 (et B005 puisqu'il s'agit de mots).

Choix du nom de fichier

Le code source produit par le désassembleur peut être dirigé sur n'importe quel périphérique classique comme, par exemple, PIO.LF, RS232.BA=9600, DSK2.XXXX, etc... Dans tous les cas, le fichier sera en format variable, 80 caractères maximum.

Comme pour les adresses, les corrections avec les touches *FCTN-S* et *FCTN-D* sont autorisées et une pression sur *BACK* provoque le retour à l'Editeur/Assembleur ou à la Mini-mémoire. Lors de la validation du nom de fichier par pression sur la touche *ENTER,* seuls les caractères situés à gauche du curseur sont pris en compte par le programme; les autres sont ignorés et disparaissent de l'écran.

Si le désassembleur est employé plusieurs fois de suite, le dernier nom de fichier utilisé réapparaît dans la zone de saisie. Dans ce cas, il suffit d'appuyer sur la touche *ENTER* pour valider le nom de fichier (si le curseur est placé sur la première position de la zone de saisie). Ceci est utile lorsque le produit du désassemblage est dirigé vers une imprimante.

Protections

Le programme est protégé contre toutes les erreurs qui peuvent se produire lors de l'entrée du nom de fichier. Les codes d'erreur générés sont les mêmes qu'avec l'Editeur/Assembleur, cela dans un but d'homogénéité. Rappelons brièvement la signification de ces codes :

0 - nom de fichier incorrect;
1 - appareil protégé contre l'écriture;
2 - mauvais attributs d'ouverture;
3 - opération illégale (par rapport aux attributs d'ouverture);
4 - dépassement de table ou tampon;
5 - tentative de lecture au-delà de la fin de fichier;
6 - toutes erreurs liées aux appareils ou supports;
7 - erreur de fichier comme, par exemple, fichier non existant ouvert en mode INPUT.

Il est évident que certaines erreurs n'apparaîtront jamais puisque le programme se charge de la gestion des fichiers.

Pendant le désassemblage proprement dit, si le "fichier source" est dirigé vers une unité de disquettes, l'état de la disquette en fonction est contrôlé en permanence. Si une erreur se produit, le message *"Disquette saturée"* s'affiche, le fichier est clos, et le désassemblage s'interrompt.

Visualisation sur écran

Le désassemblage peut être dirigé vers l'écran en donnant "E" comme nom de fichier. Si d'autres caractères suivent la lettre "E", ils sont ignorés.

L'affichage sur l'écran se fait sur 40 colonnes et par pages de 24 lignes. Pour visualiser la page suivante, il faut appuyer sur la barre d'espacement.
Le désassemblage s'arrête si la touche BACK est enfoncée. La même touche autorise aussi un retour au menu, lorsque le programme a terminé son travail.

Options

Adresses et valeurs

Cette option permet d'afficher, à côté de chaque code mnémonique, les adresses des instructions, le code machine et, si nécessaire, la ou les adresses utilisées en cas d'adressage symbolique ou symbolique indexé.

Un fichier généré avec ce complément d'information est plus pratique, car il permet au programmeur de déterminer la validité de certaines instructions par rapport au programme.

En effet, le désassembleur ne peut pas faire la différence entre une zone de données et une zone comportant des instructions. Par exemple, si le programme rencontre le code 05C7, il produira l'instruction source "INCT R7", même si cette adresse contient en réalité une valeur entrée avec la directive *"DATA"*. Le problème est le même pour les adresses contenant du texte. Il faut toutefois noter que, si le fichier source est réassemblé, le fichier objet obtenu sera correct.

Lorsque le désassembleur produit la liste des adresses et valeurs, le fichier source occupe environ trois fois plus de place. Si cela n'a pas d'importance pour un listing sur imprimante, un fichier sur disquette peut très vite prendre des proportions considérables. Il vous appartiendra donc de déterminer, en fonction des circonstances, si la liste des adresses et valeurs est requise.

La réponse par défaut à la question "Adresses et valeurs ?", lors de la première exécution du programme, est "Oui". Pour les exécutions suivantes, la réponse par défaut est celle donnée pendant la précédente séance. Une pression sur la touche *"BACK"* (encore elle !) autorise un retour au menu.

Adresses symboliques

Un programme écrit en assembleur, lorsqu'il a été réalisé par une tierce personne, n'est pas toujours facile à comprendre. Si les branchements et sauts sont indiqués en adresses réelles, la compréhension du logiciel est encore moins aisée.

Cette seconde option permet la création d'une table des adresses symboliques. Ainsi, la lecture du fichier généré par le désassembleur devient beaucoup plus facile. Il est en effet plus agréable d'obtenir des instructions comme "JNE S043" ou "BL @S200" plutôt que "JNE >A890" ou "BL @>B098".

Le programme peut stocker jusqu'à 500 adresses symboliques (S000 à S499). Si le nombre de branchements ou de sauts dépasse 500 (ce qui est peu probable), le désassemblage se poursuit normalement mais avec des adresses réelles. De même, si le programme rencontre des branchements vers des adresses situées en dehors de l'espace mémoire à désassembler, il affichera des adresses réelles.

La création de cette table se traduit par une période pendant laquelle le programme donne l'impression de s'être "planté". Le temps nécessaire pour réaliser cette tâche est directement proportionnel à la longueur de la zone de mémoire à traiter (environ 1,3 secondes par Ko).

Si le fichier source doit être réassemblé, les branchements et sauts symboliques peuvent être source de problèmes. En effet, nous avons vu précédemment que les zones de données sont parfois prises pour des instructions, ce qui laisse penser que le programme peut aussi générer des branchements inutiles. Pour éviter ce type de désagrément, il suffit de répondre "Non" à la question "Adresses symboliques ?".

En ce qui concerne les réponses par défaut, les règles sont les mêmes que pour la question "Adresses et valeurs ?".

Remise à zéro de la table

Le fichier source produit par le désassembleur est toujours plus volumineux qu'un fichier créé avec l'éditeur Texas Instruments (à cause des données transformées en instructions). Il est donc parfois nécessaire de désassembler un programme par secteurs, pour ensuite le réassembler avec la directive *"COPY"*.

Les adresses symboliques ne devant pas être dupliquées, il ne faut pas réinitialiser la table des symboles. Ainsi, si la dernière adresse du premier segment de programme est baptisée "S078", la première adresse du deuxième segment s'appelera "S079". Si la table est réinitialisée avant le désassemblage du second segment, vous verrez apparaître une impressionnante série de *"MULTIPLE SYMBOLS"* au moment du réassemblage.

La question "Remise à zéro table ?" n'est pas posée lors de la première exécution du programme. Elle est aussi "court-circuitée" si vous avez répondu "Non" à la question "Adresses symboliques ?".

Le désassembleur est enregistré sur la cassette d'accompagnement de ce numéro sous la forme d'une image mémoire. Il doit donc être chargé à l'aide de l'option "RUN PROGRAM FILE" de l'Éditeur/Assembleur. Pour information, signalons que ce programme se place en mémoire à partir de l'adresse >A000 et qu'il occupe environ 5 Ko.

Un désassembleur sur cassette n'est pas vraiment pratique puisqu'il est impossible de modifier l'adresse de chargement. De plus, si vous ne possédez pas le module Editeur/Assembleur, vous ne pouvez pas utiliser ce programme. Nous pouvons vous envoyer, contre 55 Francs, une disquette contenant :

- le fichier source listé pages suivantes (DESS);
- un code objet utilisable à partir de l'Editeur/Assembleur ou de la mini-mémoire (avec extension 32 Ko); ce fichier se charge à partir de l'adresse >2700 (DA2700);
- le même code, mais prévu pour un chargement à partir de l'adresse >A000 (DAA000);
- un code objet utilisable à partir du Basic Etendu(avec extension 32 Ko), et dont le chargement s'effectue à partir de l'adresse >2700 (CALL LOAD "DSKx.DB2700");
- le même code que le précédent, mais avec chargement à partir de l'adresse >A000 (CALL LOAD "DSKx.DBA000").

■

Exemple de Fichier source obtenu sur une imprimante matricielle Apple.

```
     MOV  R11,@>2F2E        2F34 C80B      2F2E
     LWPI >2EE0             2F38 02E0 2EE0
     BL   @S000             2F3C 06A0 2F9A
     LI   R0,>07F4          2F40 0200 07F4
     BLWP @>211C            2F44 0420 211C
     LI   R0,>0AF0          2F48 0200 0AF0
     LI   R1,>2850          2F4C 0201 2850
     LI   R2,>0018          2F50 0202 0018
     BLWP @>2110            2F54 0420 2110
     LI   R0,>0380          2F58 0200 0380
     LI   R1,>F000          2F5C 0201 F000
S001 BLWP @>210C            2F60 0420 210C
     INC  R0                2F64 0580
     CI   R0,>03A0          2F66 0280 03A0
     JNE  S001              2F6A 16FA
     LI   R0,>02FF          2F6C 0200 02FF
     LI   R1,>2000          2F70 0201 2000
S002 BLWP @>210C            2F74 0420 210C
     DEC  R0                2F78 0600
     JOC  S002              2F7A 18FC
     LI   R2,>297C          2F7C 0202 297C
     LI   R3,>298E          2F80 0203 298E
     LI   R4,>2736          2F84 0204 2736
S003 MOV  *R2+,R0           2F88 C032
     C    R2,R3             2F8A 80C2
     JEQ  >2FAE             2F8C 1310
S004 MOVB *R4+,R1           2F8E D074
```

```
      JLT  S003                2F90 11FB
      BLWP @>210C              2F92 0420 210C
      INC  R0                  2F96 0580
      JMP  S004                2F98 10FA
S000  LI   R0,>2AA0            2F9A 0200 2AA0
      LI   R1,>2EBA            2F9E 0201 2EBA
```

```
S005  SETO *R0+                2FA2 0730
      C    R0,R1               2FA4 8040
      JNE  S005                2FA6 16FD
      CLR  @>2F30              2FA8 04E0 2F30
      B    *R11                2FAC 045B
```

```
        DEF  DESA
        REF  VWTR,VSBW,VMBW,VSBR,VMBR,KSCAN,DSRLNK
        AORG >2700
DSK     DATA >0012,>1000,>5000,>0000,>0000
NFICH   DATA >2020,>2020,>2020,>2020,>2020,>2020,>2020,>2020
        DATA >2020,>2020,>2020,>2020,>2020,>2020,>2020,>2020
        DATA >2020,>2020,>2020,>2020,>2020,>2020
TEXTE1  TEXT -'* DÉSASSEMBLEUR * '
        TEXT -'_ 1984  99 Magazine '
        TEXT -'Adresses a desassembler ? '
        TEXT -'Premiere adresse ? : '
        TEXT -'Derniere adresse ? : '
        TEXT -'Fichier : '
        TEXT -'Adresses et valeurs ? '
        TEXT -'Adresses symboliques? '
TEXTE5  TEXT -'Remise a zero table ? '
TEXTE4  TEXT -'Disquette saturee '
TEXTE2  TEXT -'Erreur code '
TEXTE3  TEXT -'<ENTER> pour continuer... '
HEXA    TEXT '0123456789ABCDEFS(R)*,@>$+2-`^'
DATA    TEXT 'DATA'
SYON    BYTE >4F
ADRZON  BYTE >4F
TESP    BYTE >20
TON     BYTE >4F
TENT    BYTE >0D
TFS     BYTE >08
TFD     BYTE >09
TBACK   BYTE >0F
TO      BYTE >4F
TN      BYTE >4E
        EVEN
CREDEF  DATA >0000,>0000,>0000,>447C
        DATA >3C42,>99A1,>A199,>423C,>0000,>007C,>4444,>447C
CODE    DATA >F000,>E000,>D000,>C000,>B000,>A000,>9000
        DATA >8000,>7000,>6000,>5000,>4000,>3C00,>3800
        DATA >3400,>3000,>2C00,>2800,>2400,>2000,>1F00
        DATA >1E00,>1D00,>1C00,>1B00,>1A00,>1900,>1800
        DATA >1700,>1600,>1500,>1400,>1300,>1200,>1100
        DATA >1000,>0B00,>0A00,>0900,>0800,>0740,>0700
        DATA >06C0,>0680,>0640,>0600,>05C0,>0580,>0540
        DATA >0500,>04C0,>0480,>0440,>0400,>03E0,>03C0
        DATA >03A0,>0380,>0360,>0340,>0300,>02E0,>02C0
        DATA >02A0,>0280,>0260,>0240,>0220,>0200
VCODE   DATA >0000,>1000,>2000,>3000,>4000,>5000,>6000
        DATA >7000,>8000,>9000,>A000,>B000,>C000,>C400
        DATA >C800,>CC00,>D000,>D400,>D800,>DC00,>E000
        DATA >E100,>E200,>E300,>E400,>E500,>E600,>E700
        DATA >E800,>E900,>EA00,>EB00,>EC00,>ED00,>EE00
        DATA >EF00,>F400,>F500,>F600,>F700,>F880,>F8C0
        DATA >F900,>F940,>F980,>F9C0,>FA00,>FA40,>FA80
        DATA >FAC0,>FB00,>FB40,>FB80,>FBC0,>FC1F,>FC3F
        DATA >FC5F,>FC7F,>FC9F,>FCBF,>FCFF,>FD1F,>FD30
        DATA >FD50,>FD70,>FD90,>FDB0,>FDD0,>FDF0
ADRECR  DATA >0028,>0067,>00C4,>0104,>0124,>0184,>01E4,>0204
MNEMO   TEXT 'SOCBSOC MOVBMOV AB  A   CB  '
        TEXT 'C   SB  S   SZCBSZC DIV MPY '
        TEXT 'STCRLDCRXOP XOR CZC COC TB  '
        TEXT 'SBZ SBO JOP JH  JL  JNO JOC '
        TEXT 'JNC JNE JGT JHE JEQ JLE JLT '
        TEXT 'JMP SRC SLA SRL SRA ABS SETO'
        TEXT 'SWPBBL  DECTDEC INCTINC INV '
        TEXT 'NEG CLR X   B   BLWPLREXCKOF'
        TEXT 'CKONRTWPRSETIDLELIMILWPISTST'
        TEXT 'STWPCI  ORI ANDIAI  LI  '
ADRETQ  BSS  >03E8
TLIGNE  BSS  >50
TCLAV   BSS  >4
ADRS    BSS  >4
REGPP   BSS  >20
REGSP   BSS  >20
TEST    DATA >E00D
WRCL    DATA >0301
DR1P    DATA 0
DRECR   DATA 0
DRADRS  DATA 0
TABLEI  DATA 0
ADRON   DATA 0
ADRRET  DATA 0
PRETQ   DATA 0
SARGPL  DATA 0
PLONG   EQU  >8356
ETAT    EQU  >837C
RTCLAV  EQU  >8375
TPAB    EQU  >1000
PAB     EQU  >F80
* affichage
DESA    MOV  R11,@ADRRET
        LWPI REGPP
        BL   @FAFT
DESAR   LI   R0,>07F4
        BLWP @VWTR
        LI   R0,>0AF0
        LI   R1,CREDEF
        LI   R2,>18
        BLWP @VMBW
        LI   R0,>0380
        LI   R1,>F000
TCOUL   BLWP @VSBW
        INC  R0
        CI   R0,>03A0
        JNE  TCOUL
        LI   R0,>02FF
        LI   R1,>2000
EFEC    BLWP @VSBW
        DEC  R0
        JOC  EFEC
        LI   R2,ADRECR
        LI   R3,ADRECR+>12
        LI   R4,TEXTE1
AFT     MOV  *R2+,R0
        C    R2,R3
        JEQ  DSAIS
AFT1    MOVB *R4+,R1
        JLT  AFT
        BLWP @VSBW
        INC  R0
        JMP  AFT1
* reinitialisation table
FAFT    LI   R0,ADRETQ
        LI   R1,ADRETQ+>03EA
VT      SETO *R0+
        C    R0,R1
        JNE  VT
        CLR  @PRETQ
        RT
* saisie adresses hexa
DSAIS   LI   R10,1
```

```
DSAIS0 BL    @SAISIE
       DATA  >0119
       BL    @CARN
       COC   R10,R3
       JEQ   DSAIS0
       MOV   R3,@ADRS
DSAIS1 BL    @SAISIE
       DATA  >0139
       BL    @CARN
       COC   R10,R3
       JEQ   DSAIS1
       MOV   R3,@ADRS+2
* verif si erreur
       C     @ADRS+2,@ADRS
       JH    SNF
       LI    R0,>0119
       CLR   R1
       LI    R2,4
ERSAI1 BLWP  @VSBW
       INC   R0
       DEC   R2
       JNE   ERSAI1
       LI    R0,>0139
       LI    R2,4
ERSAI2 BLWP  @VSBW
       INC   R0
       DEC   R2
       JNE   ERSAI2
       JMP   DSAIS
* saisie nom fichier
SNF    LI    R0,>02AA
       LI    R1,>2000
SNF0   BLWP  @VSBW
       INC   R0
       CI    R0,>02FD
       JNE   SNF0
       LI    R0,>018E
       LI    R1,NFICH
       LI    R2,>2C
       BLWP  @VMBW
       LI    R7,NFICH
       CLR   R2
SNF1   LI    R3,>100
SNF2   BLWP  @KSCAN
       MOVB  @ETAT,R4
       JEQ   SNFCL
       CB    @RTCLAV,@TENT
       JEQ   SNFE
       CB    @RTCLAV,@TBACK
       JNE   SNF5
       B     @RETOUR
SNF5   CB    @RTCLAV,@TFS
       JNE   SNF3
       MOV   R2,R2
       JEQ   SNF2
       MOVB  *R7,R1
       BLWP  @VSBW
       DEC   R2
       DEC   R7
       DEC   R0
       JMP   SNF2
SNF3   CB    @RTCLAV,@TFD
       JNE   SNF4
       CI    R2,>2B
       JEQ   SNF2
       MOVB  *R7+,R1
       BLWP  @VSBW
       INC   R2
       INC   R0
       JMP   SNF2
SNF4   CI    R2,>2B
```

```
       JEQ   SNF2
       MOVB  @RTCLAV,R1
       MOVB  R1,*R7+
       BLWP  @VSBW
       INC   R0
       INC   R2
       JMP   SNF2
SNFCL  DEC   R3
       JNE   SNF2
       MOV   R6,R6
       JEQ   SNFCL0
       CLR   R6
       MOVB  @HEXA+>1C,R1
       BLWP  @VSBW
       JMP   SNF1
SNFCL0 SETO  R6
       MOVB  *R7,R1
       BLWP  @VSBW
       JMP   SNF1
SNFE   MOVB  *R7,R1
       BLWP  @VSBW
       MOV   R2,R2
       JNE   SNFE1
       CB    @NFICH,@TESP
       JNE   SAONX
       JMP   SNF2
SNFE1  SWPB  R2
       MOVB  R2,@DSK+9
       LI    R6,>2000
       SWPB  R2
SNFE0  MOVB  R6,@NFICH(R2)
       INC   R2
       CI    R2,>2C
       JNE   SNFE0
       LI    R0,>018E
       LI    R1,NFICH
       LI    R2,>2C
       BLWP  @VMBW
* saisies options
SAONX  CB    @NFICH,@HEXA+14
       JNE   SAON
       SETO  @DRECR
       JMP   SAON1
SAON   CLR   @DRECR
SAON1  LI    R0,>01FA
       LI    R3,>AAAA
       LI    R2,TON
       BL    @ON
       MOVB  @TON,R1
       BLWP  @VSBW
       CB    R1,@TO
       JNE   SAON0
       SETO  @ADRON
       JMP   ASON
SAON0  CLR   @ADRON
ASON   LI    R0,>021A
       LI    R2,SYON
       BL    @ON
       MOVB  @SYON,R1
       BLWP  @VSBW
       CB    R1,@TO
       JNE   ASON0
       SETO  @DRADRS
       JMP   RZON
ASON0  CLR   @DRADRS
       BL    @FAFT
       JMP   OUVF
RZON   MOV   @DR1P,R1
       JNE   RZON0
       SETO  @DR1P
       JMP   OUVF
```

```
RZON0  LI    R0,>0224
       LI    R2,TEXTE5
RZON1  MOVB  *R2+,R1
       JLT   RZON2
       BLWP  @VSBW
       INC   R0
       JMP   RZON1
RZON2  LI    R0,>023A
       LI    R2,ADRZON
       BL    @ON
       MOVB  @ADRZON,R1
       BLWP  @VSBW
       CB    R1,@TO
       JNE   OUVF
       BL    @FAFT
* ouverture fichier
OUVF   MOV   @DRECR,R1
       JEQ   OUVF6
       LI    R0,2
       MOV   R0,@TABLEI
       JMP   TABETQ
OUVF6  LI    R0,PAB
       LI    R1,DSK
       LI    R2,>36
       BLWP  @VMBW
       LI    R0,PAB+9
       MOV   R0,@PLONG
       CLR   R1
       MOVB  @>9802,@SARGPL
       MOVB  @>9802,@SARGPL+1
       DEC   @SARGPL
       MOVB  R1,@ETAT
       BLWP  @DSRLNK
       DATA  >0008
       JNE   TABETQ
       LI    R0,PAB+1
       CLR   R1
       BLWP  @VSBR
       ANDI  R1,>E000
OUVFE  MOV   R1,R3
       LI    R0,>02AA
       LI    R2,TEXTE2
OUVF2  MOVB  *R2+,R1
       JLT   OUVF4
       BLWP  @VSBW
       INC   R0
       JMP   OUVF2
OUVF4  LI    R0,>02E4
       LI    R2,TEXTE3
OUVF5  MOVB  *R2+,R1
       JLT   OUVF1
       BLWP  @VSBW
       INC   R0
       JMP   OUVF5
OUVF1  SRL   R3,13
       MOVB  @HEXA(R3),R1
       LI    R0,>02B6
       BLWP  @VSBW
       LI    R2,>A000
BDELAI DEC   R2
       JNE   BDELAI
BKSCAN BLWP  @KSCAN
       CB    @RTCLAV,@TENT
       JNE   BKSCAN
       B     @SNF
* creation table etq
TABETQ MOV   @DRADRS,R1
       JEQ   DES
       MOV   @ADRS,R1
       MOV   @PRETQ,R3
TAB3   C     R1,@ADRS+2
```

```
        JLE   TAB14
        MOV   R3,@PRETQ
        JMP   DES
TAB14   MOV   *R1+,R4
        CLR   R8
TAB1    COC   @CODE(R8),R4
        JNE   TAB6
        CZC   @VCODE(R8),R4
        JEQ   TAB5
TAB6    INCT  R8
        CI    R8,>8A
        JNE   TAB1
        JMP   TAB3
TAB5    CI    R8,>2E
        JLT   TAB7
        CI    R8,>46
        JGT   TAB2
        ANDI  R4,>FF
        JEQ   TAB3
        BL    @CADRR
        BL    @VEXADR
        JMP   TAB3
TAB2    CI    R8,>56
        JEQ   TAB4
        CI    R8,>68
        JEQ   TAB4
        CI    R8,>6A
        JNE   TAB7
TAB4    ANDI  R4,>3F
        CI    R4,>20
        JNE   TAB3
        MOV   *R1+,R10
        BL    @VEXADR
        JMP   TAB3
TAB7    CI    R8,>16
        JGT   TAB9
        MOV   R4,R5
        ANDI  R5,>30
        CI    R5,>20
        JNE   TAB8
        INCT  R1
TAB8    MOV   R4,R5
        ANDI  R5,>C00
        CI    R5,>800
        JNE   TAB3
        INCT  R1
        JMP   TAB3
TAB9    CI    R8,>26
        JGT   TAB10
TAB12   MOV   R4,R5
        ANDI  R5,>30
        CI    R5,>20
        JNE   TAB3
TAB13   INCT  R1
        JMP   TAB3
TAB10   CI    R8,>50
        JLT   TAB3
        CI    R8,>6A
        JGT   TAB11
        JMP   TAB12
TAB11   CI    R8,>78
        JLT   TAB3
        CI    R8,>7C
        JLT   TAB13
        CI    R8,>7E
        JGT   TAB13
        JMP   TAB3
* boucle desassemblage
DES     MOV   @DRECR,R1
        JEQ   DES30
        LI    R0,>F000
        MOVB  R0,@>83D4
        LI    R0,>01F0
        BLWP  @VWTR
        LI    R0,>03BF
        LI    R1,>2000
DES31   BLWP  @VSBW
        DEC   R0
        JOC   DES31
DES30   MOV   @ADRS,R1
DES0    C     R1,@ADRS+2
        JLE   DES8
        B     @DESF
DES8    LI    R10,TLIGNE
        LI    R9,>2000
DES1    MOVB  R9,*R10+
        CI    R10,TLIGNE+>50
        JNE   DES1
        MOV   @ADRON,R10
        JEQ   DES23
        MOV   R1,R8
        BL    @AFH
        DATA  TLIGNE+>22
        MOV   *R1,R8
        BL    @AFH
        DATA  TLIGNE+>27
DES23   CLR   R10
DES4    C     R1,@ADRETQ(R10)
        JNE   DES2
        MOV   R10,R9
        SRL   R9,1
        BL    @AFETQ
        DATA  TLIGNE
        JMP   DES3
DES2    INCT  R10
        CI    R10,>03E8
        JNE   DES4
DES3    CLR   R2
        MOV   *R1+,R4
DES6    COC   @CODE(R2),R4
        JEQ   DES5
        INCT  R2
        CI    R2,>8A
        JNE   DES6
DES18   LI    R10,TLIGNE+>5
        LI    R9,DATA
DES7    MOVB  *R9+,*R10+
        CI    R9,DATA+>4
        JNE   DES7
        MOV   R4,R8
        MOVB  @HEXA+>17,@TLIGNE+>A
        BL    @AFH
        DATA  TLIGNE+>B
        BLWP  @FVI
        JMP   DESN
DES5    CZC   @VCODE(R2),R4
        JNE   DES18
        MOV   R2,R10
        SLA   R10,1
        LI    R9,TLIGNE+>5
DES22   MOVB  @MNEMO(R10),*R9+
        INC   R10
        CI    R9,TLIGNE+>9
        JNE   DES22
        CI    R2,>16
        JGT   DES9
        BLWP  @SO
        BLWP  @FID
        JMP   DESN
DES9    CI    R2,>1A
        JGT   DES10
DES12   BLWP  @SO
        BLWP  @FIX
        JMP   DESN
DES10   CI    R2,>1E
        JGT   DES11
        BLWP  @SO
        BLWP  @FIV
        JMP   DESN
DES11   CI    R2,>20
        JEQ   DES12
        CI    R2,>26
        JGT   DES13
        BLWP  @SO
        BLWP  @FIX
        JMP   DESN
DES13   CI    R2,>2C
        JGT   DES14
        BLWP  @FIICR
        JMP   DESN
DES14   CI    R2,>46
        JGT   DES15
        BLWP  @FIIJ
        JMP   DESN
DES15   CI    R2,>4E
        JGT   DES16
        BLWP  @FV
        JMP   DESN
DES16   CI    R2,>6A
        JGT   DES17
        BLWP  @SO
        BLWP  @FVI
        JMP   DESN
DES17   CI    R2,>76
        JGT   DES19
        MOV   R4,R8
        ANDI  R8,>1F
        JNE   DES18
        BLWP  @FVI
        JMP   DESN
DES19   CI    R2,>88
        JGT   DES18
        MOV   R4,R8
        ANDI  R8,>10
        JNE   DES18
        CI    R2,>7A
        JGT   DES20
        BLWP  @FVIII2
        JMP   DESN
DES20   CI    R2,>7E
        JGT   DES21
        BLWP  @FVIII1
        JMP   DESN
DES21   BLWP  @FVIII3
DESN    B     @DES0
* sp format 8 (AI...)
FVIII3  DATA  REGSP
        DATA  $+2
        LI    R1,TLIGNE+>A
        MOV   @REGPP+>8,R2
        MOV   R2,R4
        ANDI  R4,>F
        BL    @AFREG
        MOVB  @HEXA+>15,*R1+
        MOV   @REGPP+2,R3
        INCT  @REGPP+2
        MOV   *R3,R8
        MOVB  @HEXA+>17,*R1+
        MOV   R1,R10
        BL    @AFH1
        MOV   @ADRON,R0
        JEQ   FVIIIX
        MOV   *R3,R8
```

```
       BL    @AFH
       DATA TLIGNE+)2C
FVIIIX BL    @ENR
       RTWP
* sp format 8 (STWP...)
FVIII2 DATA REGSP
       DATA $+2
       MOV   @REGPP+2,R3
       INCT @REGPP+2
       MOV   *R3,R8
       MOVB @HEXA+)17,@TLIGNE+)A
       BL    @AFH
       DATA TLIGNE+)B
       MOV   @ADRON,R0
       JEQ   FVIIIY
       MOV   *R3,R8
       BL    @AFH
       DATA TLIGNE+)2C
FVIIIY BL    @ENR
       RTWP
* sp format 8 (LWPI...)
FVIII1 DATA REGSP
       DATA $+2
       MOV   @REGPP+4,R4
       ANDI R4,)F
       LI    R1,TLIGNE+)A
       BL    @AFREG
       BL    @ENR
       RTWP
* sp format 9
FIX    DATA REGSP
       DATA $+2
       MOVB @HEXA+)15,*R1+
       MOV   R2,R4
       SRL   R4,6
       ANDI R4,)F
       MOV   @REGPP+4,R0
       CI    R0,)20
       JNE   FIX0
       BL    @AFREG1
       JMP   FIX1
FIX0   BL    @AFREG
FIX1   BL    @ENR
       RTWP
* sp format 4
FIV    DATA REGSP
       DATA $+2
       MOVB @HEXA+)15,*R1+
       MOV   R2,R4
       SRL   R4,6
       ANDI R4,)F
       BL    @AFREG1
       BL    @ENR
       RTWP
* sp format 2 CRU
FIICR  DATA REGSP
       DATA $+2
       CLR   R9
       MOV   @REGPP+8,R9
       ANDI R9,)FF
       LI    R7,TLIGNE+)A
       CI    R9,)80
       JLT   FIICR2
       MOVB @HEXA+)1B,*R7+
       AI    R9,)FF00
       ABS   R9
FIICR2 CLR   R8
       LI    R10,)64
       DIV   R10,R8
       MOV   R8,R8
       JEQ   FIICR0
```

```
       MOVB @HEXA(R8),*R7+
FIICR0 CLR   R8
       LI    R10,)A
       DIV   R10,R8
       MOV   R8,R8
       JEQ   FIICR1
       MOVB @HEXA(R8),*R7+
FIICR1 MOVB @HEXA(R9),*R7
       BL    @ENR
       RTWP
* sp adr. gener. source
S0     DATA REGSP
       DATA $+2
       LI    R1,TLIGNE+)A
       MOV   @REGPP+8,R2
       MOV   R2,R3
       ANDI R3,)30
       JNE   S00
       MOV   R2,R4
       ANDI R4,)F
       BL    @AFREG
       JMP   S0F
S00    LI    R5,)FFCF
       MOV   R2,R6
       SZC   R5,R6
       CI    R6,)10
       JNE   S01
       MOVB @HEXA+)14,*R1+
       MOV   R2,R4
       ANDI R4,)F
       BL    @AFREG
       JMP   S0F
S01    CI    R6,)30
       JNE   S02
       MOVB @HEXA+)14,*R1+
       MOV   R2,R4
       ANDI R4,)F
       BL    @AFREG
       MOVB @HEXA+)19,*R1+
       JMP   S0F
S02    MOV   @REGPP+2,R3
       INCT @REGPP+2
       MOV   *R3,R8
       MOV   @REGPP+4,R7
       CI    R7,)56
       JEQ   S0XX
       CI    R7,)68
       JEQ   S0XX
       CI    R7,)6A
       JNE   S04
S0XX   C     R8,@ADRS
       JL    S04
       C     R8,@ADRS+2
       JH    S04
       CLR   R9
S06    C     R8,@ADRETQ(R9)
       JEQ   S05
       INCT R9
       CI    R9,)03E8
       JNE   S06
       JMP   S04
S05    MOVB @HEXA+)16,*R1+
       SRL   R9,1
       BL    @AFETQ
       DATA TLIGNE+)B
       MOV   @ADRON,R0
       JEQ   S0F
       MOV   *R3,R8
       BL    @AFH
       DATA TLIGNE+)2C
       JMP   S0F
```

```
S04    MOVB @HEXA+)16,*R1+
       MOVB @HEXA+)17,*R1+
       MOV   R1,R10
       BL    @AFH1
       MOV   R10,R1
       MOV   @ADRON,R0
       JEQ   S07
       BL    @AFH
       DATA TLIGNE+)2C
S07    MOV   R2,R4
       ANDI R4,)F
       JEQ   S0F
       MOVB @HEXA+)11,*R1+
       BL    @AFREG
       MOVB @HEXA+)13,*R1+
S0F    RTWP
* sp format 1 destination
FID    DATA REGSP
       DATA $+2
       MOVB @HEXA+)15,*R1+
       MOV   R2,R3
       ANDI R3,)C00
       JNE   FID0
       MOV   R2,R4
       SRL   R4,6
       ANDI R4,)F
       BL    @AFREG
       JMP   FIF
FID0   LI    R5,)F3FF
       MOV   R2,R6
       SZC   R5,R6
       CI    R6,)400
       JNE   FID1
       MOVB @HEXA+)14,*R1+
       MOV   R2,R4
       SRL   R4,6
       ANDI R4,)F
       BL    @AFREG
       JMP   FIF
FID1   CI    R6,)C00
       JNE   FID2
       MOVB @HEXA+)14,*R1+
       MOV   R2,R4
       SRL   R4,6
       ANDI R4,)F
       BL    @AFREG
       MOVB @HEXA+)19,*R1+
       JMP   FIF
FID2   MOV   @REGPP+2,R3
       INCT @REGPP+2
       MOV   *R3,R8
       MOVB @HEXA+)16,*R1+
       MOVB @HEXA+)17,*R1+
       MOV   R1,R10
       BL    @AFH1
       MOV   R10,R1
       MOV   @ADRON,R0
       JEQ   FID3
       BL    @AFH
       DATA TLIGNE+)31
FID3   MOV   R2,R4
       ANDI R4,)3C0
       JEQ   FIF
       MOVB @HEXA+)11,*R1+
       SRL   R4,6
       BL    @AFREG
       MOVB @HEXA+)13,*R1+
FIF    BL    @ENR
       RTWP
* sp format 2 adr. relative
FIIJ   DATA REGSP
```

```
        DATA $+2
        MOV  @REGPP+8,R0
        ANDI R0,>FF
        JNE  FIIJ0
        MOVB @HEXA+>18,@TLIGNE+>A
        MOVB @HEXA+>19,@TLIGNE+>B
        MOVB @HEXA+>1A,@TLIGNE+>C
        JMP  FIIJF
FIIJ0   MOV  @REGPP+2,R1
        BL   @CADRR
        CLR  R9
FIIJ1   C    R10,@ADRETQ(R9)
        JEQ  FIIJ2
        INCT R9
        CI   R9,>3E8
        JNE  FIIJ1
        MOVB @HEXA+>17,@TLIGNE+>A
        MOV  R10,R8
        BL   @AFH
        DATA TLIGNE+>B
        JMP  FIIJF
FIIJ2   SRL  R9,1
        BL   @AFETQ
        DATA TLIGNE+>A
FIIJF   BL   @ENR
        RTWP
* sp format 5
FV      DATA REGSP
        DATA $+2
        MOV  @REGPP+8,R4
        ANDI R4,>F
        LI   R1,TLIGNE+>A
        BL   @AFREG
        MOVB @HEXA+>15,*R1+
        MOV  @REGPP+8,R4
        SRL  R4,4
        ANDI R4,>F
        BL   @AFREG1
        BL   @ENR
        RTWP
* sp format 6
FVI     DATA REGSP
        DATA $+2
        BL   @ENR
        RTWP
* sp saisie nombres hexa
SAISIE  MOV  *R11+,R5
SAIS0   MOV  R5,R0
        LI   R1,>2000
        LI   R2,5
SAIS1   BLWP @VSBW
        INC  R0
        DEC  R2
        JNE  SAIS1
        MOV  R5,R0
        LI   R7,TCLAV
        CLR  R2
        MOV  R2,@TCLAV
        MOV  R2,@TCLAV+2
SAIS2   LI   R3,>100
SAIS3   BLWP @KSCAN
        MOVB @ETAT,R4
        JEQ  SAISCL
        CB   @RTCLAV,@TBACK
        JNE  SAIS8
        B    @RETOUR
SAIS8   CB   @RTCLAV,@TENT
        JNE  SAIS4
        CI   R2,3
        JNE  SAIS3
        MOVB @TCLAV+3,R1
        BLWP @VSBW
        LI   R1,3
SAISV   MOVB @TCLAV(R1),R8
        JEQ  SAIS0
        DEC  R1
        JOC  SAISV
        RT
SAIS4   CB   @RTCLAV,@TFS
        JNE  SAIS5
        MOV  R2,R2
        JEQ  SAIS3
        MOVB *R7,R1
        BLWP @VSBW
        DEC  R2
        DEC  R7
        DEC  R0
        JMP  SAIS3
SAIS5   CB   @RTCLAV,@TFD
        JNE  SAIS6
        CI   R2,3
        JEQ  SAIS3
        MOVB *R7+,R1
        BLWP @VSBW
        INC  R0
        INC  R2
        JMP  SAIS3
SAIS6   CB   @RTCLAV,@HEXA
        JLT  SAIS0
        CB   @RTCLAV,@HEXA+>9
        JGT  SAIS7
        JMP  SAISA
SAIS7   CB   @RTCLAV,@HEXA+>A
        JLT  SAIS0
        CB   @RTCLAV,@HEXA+>F
        JGT  SAIS0
SAISA   MOVB @RTCLAV,R1
        MOVB R1,*R7
        BLWP @VSBW
        CI   R2,3
        JEQ  SAIS3
        INC  R0
        INC  R2
        INC  R7
        JMP  SAIS3
SAISCL  DEC  R3
        JNE  SAIS3
        MOV  R6,R6
        JEQ  SAISC0
        CLR  R6
        MOVB @HEXA+>1C,R1
        BLWP @VSBW
        JMP  SAIS2
SAISC0  SETO R6
        MOVB *R7,R1
        BLWP @VSBW
        JMP  SAIS2
* sp conver. ASCII/binaire
CARN    LI   R1,TCLAV
        CLR  R0
        CLR  R3
        LI   R4,>4C
CARN2   CB   *R1,@HEXA+9
        JLE  CARN1
        AB   @CODE(R4),*R1
CARN1   CLR  R2
        MOVB *R1+,R2
        SLA  R2,4
        SRC  R2,0
        SOC  R2,R3
        AI   R0,4
        CI   R0,16
        JNE  CARN2
        RT
* sp affichage registres
AFREG   MOVB @HEXA+>12,*R1+
AFREG1  CLR  R3
        LI   R10,>A
        DIV  R10,R3
        MOV  R3,R3
        JEQ  AFREG0
        MOVB @HEXA(R3),*R1+
AFREG0  MOVB @HEXA(R4),*R1+
        RT
* sp calcul adr. relatives
CADRR   MOV  R1,R10
        CLR  R9
        MOVB @REGPP+9,R9
        SRA  R9,7
        A    R9,R10
        RT
* sp verif. si adresse
* dans table etq
VEXADR  CI   R3,>03E8
        JEQ  VEXF
        C    R10,@ADRS
        JL   VEXF
        C    R10,@ADRS+2
        JH   VEXF
        CLR  R9
VEX0    C    R10,@ADRETQ(R9)
        JEQ  VEXF
        INCT R9
        CI   R9,>3E8
        JNE  VEX0
        MOV  R10,@ADRETQ(R3)
        INCT R3
VEXF    RT
* sp affichage etq
AFETQ   MOV  *R11+,R7
        MOVB @HEXA+>10,*R7+
        CLR  R8
        LI   R10,>64
        DIV  R10,R8
        MOVB @HEXA(R8),*R7+
        CLR  R8
        LI   R10,>A
        DIV  R10,R8
        MOVB @HEXA(R8),*R7+
        MOVB @HEXA(R9),*R7
        RT
* sp affichage nombres hexa
AFH     MOV  *R11+,R10
AFH1    LI   R0,>C
AFH0    MOV  R8,R9
        SRC  R9,0
        ANDI R9,>F
        MOVB @HEXA(R9),*R10+
        MOV  R0,R0
        JEQ  AFHF
        DECT R0
        DECT R0
        JMP  AFH0
AFHF    RT
* sp enregistrement ligne
ENR     MOV  @DRECR,R1
        JNE  AFECR
ENR2    LI   R1,>50
        LI   R2,TLIGNE+>4F
        LI   R3,>2000
ENR1    CB   *R2,R3
        JNE  ENR0
        DEC  R1
```

```
        DEC  R2
        JMP  ENR1
ENR0    LI   R0,PAB+5
        SWPB R1
        BLWP @VSBW
        LI   R0,TPAB
        LI   R1,TLIGNE
        LI   R2,>50
        BLWP @VMBW
        MOVB @WRCL,R1
        LI   R0,PAB
        BLWP @VSBW
        LI   R6,PAB+9
        MOV  R6,@PLONG
        LI   R1,>1200
        LI   R0,PAB+1
        BLWP @VSBW
        CLR  R0
        MOVB R0,@ETAT
        BLWP @DSRLNK
        DATA 8
        LI   R0,PAB+1
        BLWP @VSBR
        ANDI R1,>E000
        JNE  DSKS
        RT
* sp affichage sur ecran
AFECR   MOV  @TABLEI,R0
        LI   R1,TLIGNE
        LI   R2,>26
        BLWP @VMBW
        AI   R0,>28
        MOV  R0,@TABLEI
        CI   R0,>03C2
        JEQ  AFECR0
        RT
AFECR0  LI   R0,2
        MOV  R0,@TABLEI
AFECR2  BLWP @KSCAN
        CB   @RTCLAV,@TBACK
        JNE  AFECR1
        LI   R0,>E000
        MOVB R0,@>83D4
        LI   R0,>01E0
        BLWP @VWTR
        LI   R0,>0300
        LI   R1,>D000
        BLWP @VSBW
        LWPI REGPP
        B    @DESAR
AFECR1  CB   @RTCLAV,@TESP
        JNE  AFECR2
        LI   R0,>0040
        LI   R1,>2000
        LI   R2,>03C0
        MOVB R0,@>8C02
        SWPB R0
        MOVB R0,@>8C02
        NOP
AFECR3  MOVB R2,@>8C00
        DEC  R2
        JNE  AFECR3
        RT
* sp si disq. saturee
DSKS    LI   R0,>02A8
        LI   R2,TEXTE4
DSKS0   MOVB *R2+,R1
        JLT  DSKS1
        BLWP @VSBW
        INC  R0
        JMP  DSKS0
DSKS1   LI   R0,>02E4
        LI   R2,TEXTE3
DSKS2   MOVB *R2+,R1
        JLT  DSKS3
        BLWP @VSBW
        INC  R0
        JMP  DSKS2
DSKS3   BLWP @KSCAN
        CB   @RTCLAV,@TENT
        JNE  DSKS3
        B    @DESAR
* fermeture fichier
DESF    MOV  @DRECR,R1
        JNE  DESF0
        MOVB @WRCL+1,R1
        LI   R0,PAB
        BLWP @VSBW
        LI   R6,PAB+9
        MOV  R6,@PLONG
        BLWP @DSRLNK
        DATA 8
        B    @DESAR
DESF0   BLWP @KSCAN
        CB   @RTCLAV,@TBACK
        JNE  DESF0
        LI   R0,>E000
        MOVB R0,@>83D4
        LI   R0,>01E0
        BLWP @VWTR
        LI   R0,>0300
        LI   R1,>D000
        BLWP @VSBW
        B    @DESAR
* vers edit/ass ou mini-m.
RETOUR  MOVB @SARGPL,@>9C02
        MOVB @SARGPL+1,@>9C02
        CLR  R0
        MOVB R0,@ETAT
        MOV  @ADRRET,R11
        RT
* sp pour saisies options
ON      LI   R4,>120
ON0     BLWP @KSCAN
        MOVB @ETAT,R5
        JEQ  ONC
        MOVB @RTCLAV,R1
        CB   R1,@TENT
        JNE  ON1
        RT
ON1     CB   R1,@TBACK
        JNE  ON2
        B    @DESAR
ON2     CB   R1,@TO
        JEQ  ONV
        CB   R1,@TN
        JNE  ON0
ONV     MOVB R1,*R2
        BLWP @VSBW
        JMP  ON0
ONC     DEC  R4
        JNE  ON0
        SRC  R3,1
        JOC  ONC0
        MOVB *R2,R1
        JMP  ONC1
ONC0    MOVB @HEXA+>1D,R1
ONC1    BLWP @VSBW
        JMP  ON
        END  DESA
```

A vous de programmer

Vous connaissez certainement la célèbre émission de télévision *"Des chiffres et des lettres"*. Une partie de ce jeu est basé sur la reconstitution d'un nombre choisi aléatoirement, à l'aide d'autres nombres choisis de la même manière.

Nous vous proposons cette fois d'écrire un programme capable de trouver la solution des jeux pendant le cours de l'émission précitée et, si possible, avant les candidats.

Des nombres... un ordinateur... cela paraît très simple mais, lorsque vous vous serez penché sur le problème, vous pourrez constater que trouver la solution n'est pas chose aisée.

Au moment de votre départ en vacances, n'oubliez pas votre TI !

Bon courage... ■

Les tours de Hanoï

Daniel Masse

Cette nouvelle version (voir numéro 1) écrite en **Basic Etendu** autorise l'utilisation du synthétiseur de paroles.

Pour ceux qui ne connaissent pas encore ce classique, nous exposons les règles du jeu.

Le "terrain de jeu" est composé de trois tours. Sur la première sont empilés des disques de diamètres différents et il faut reconstituer cette pile sur l'une des deux autres tours.
Pour ce faire, les disques doivent être déplacés un par un d'une tour à l'autre, sans jamais poser un disque sur un autre plus petit.

Vous pouvez choisir le nombre de disques et ainsi faire varier le niveau du jeu. Si la solution du problème est rapidement trouvée pour trois disques, il n'en va pas de même si votre choix se porte sur sept disques, car les choses se compliquent sérieusement.

■

```
100 ! *********************
110 ! * Les tours de Hanoi *
120 ! *********************
130 ! *    Basic Etendu    *
140 ! *********************
150 !
160 CALL CLEAR
170 GOSUB 1150
180 GOSUB 1260
190 GOSUB 1420
200 GOSUB 1260
210 GOSUB 1150
220 DIM T(7,3)
230 CALL CHAR(128,"8040707F3F3F1F")
240 CALL CHAR(136,"002418FFFFFFFF")
250 CALL CHAR(129,"01020EFEFCFCF8")
260 CALL CHAR(130,"08040707030301")
270 CALL CHAR(137,"004281FFFFFFFF")
280 CALL CHAR(106,"004080FFFFFFFF")
290 CALL CHAR(131,"1020E0E0C0C080")
300 CALL CHAR(112,"0105070301010101")
310 CALL CHAR(113,"FFFFFFFFFFFFFFFF")
320 CALL CHAR(114,"80A0E0C080808080")
330 CALL CHAR(120,"030F3FFFFFFFFFFF")
340 CALL CHAR(121,"00000000030F3FFF")
350 CALL CHAR(122,"C0F0FCFFFFFFFFFF")
360 CALL CHAR(123,"00000000C0F0FCFF")
370 CALL CHAR(124,"FFFFFFFFFFFFFFFF")
380 CALL SCREEN(6)
390 CALL COLOR(13,11,1,14,10,1,11,13,1,12
,3,1)
400 FOR I=0 TO 2
410 FOR J=0 TO 1
420 CALL HCHAR(13+J,6+I*9,112)
430 CALL HCHAR(13+J,7+I*9,113)
440 CALL HCHAR(13+J,8+I*9,114)
450 CALL HCHAR(15+J,6-2*J+9*I,120)
460 CALL HCHAR(15+J,5-2*J+9*I,121)
470 CALL HCHAR(15+J,8+2*J+9*I,122)
480 CALL HCHAR(15+J,9+2*J+9*I,123)
490 NEXT J
500 CALL HCHAR(15,7+9*I,124)
510 CALL HCHAR(16,5+9*I,124,5)
520 NEXT I
530 CALL HCHAR(17,2,124,29)
540 A$(1)=CHR$(128)&CHR$(136)&CHR$(136)&C
HR$(136)&CHR$(136)&CHR$(136)&CHR$(136)&
CHR$(136)&CHR$(129)
550 A$(2)=CHR$(130)&CHR$(137)&CHR$(137)&C
HR$(137)&CHR$(137)&CHR$(137)&CHR$(137)&
CHR$(137)&CHR$(131)
560 FOR I=1 TO 3
570 A$(2*I+1)=SEG$(A$(1),1,8-2*I)&CHR$(12
9)
580 A$(2*I+2)=SEG$(A$(2),1,8-2*I)&CHR$(13
1)
590 NEXT I
600 REM JEU
610 DISPLAY AT(20,0):"COMBIEN DE DISQUES
(3 A 7) ?":" ":" ":" ":" " :: C=0 :: FO
R I=1 TO 7 :: FOR J=1 TO 3 :: T(I,J)=0 ::
 NEXT J :: NEXT I
620 IF SS THEN CALL SAY("#READY TO START#
.NUMBER OF DISKETTE")
630 ACCEPT AT(20,28):NBD
640 FOR I=6 TO 12 :: DISPLAY AT(I,1):" "
:: NEXT I :: IF NBD<3 OR NBD>7 THEN 610
650 FOR I=1 TO NBD
660 DISPLAY AT(13-I,1+INT((I-1)/2)):A$(I)
670 T(I,1)=I
680 NEXT I
690 DISPLAY AT(20,1):"TOUR DE DEPART ?"
700 CALL KEY(0,K,S):: IF S=0 THEN 690 ELS
E TD=VAL(CHR$(K))
710 IF T(1,TD)=0 AND SS=1 THEN CALL SAY("
#WHAT WAS THAT#")
720 IF T(1,TD)=0 THEN DISPLAY AT(22,1):"C
ETTE TOUR EST VIDE..." :: GOTO 700
730 IF (TD<1 OR TD>3)AND SS=1 THEN CALL S
AY("#WHAT WAS THAT#")
740 IF TD<1 OR TD>3 THEN DISPLAY AT(22,1)
:"COMPTEZ JUSQU'A 3..." :: GOT 570
750 DISPLAY AT(22,1):" "
760 DISPLAY AT(20,16):TD
770 DISPLAY AT(21,9):"D'ARRIVEE ?"
780 CALL KEY(0,K,S):: IF S=0 THEN 770 ELS
E TA=VAL(CHR$(K))
790 IF TA=TD AND SS=1 THEN CALL SAY("#WHA
T WAS THAT#")
800 IF TA=TD THEN DISPLAY AT(22,1):"LE SU
R-PLACE VOUS AMUSE ?" :: GOTO 770
810 IF (TA<1 OR TA>3)AND SS=1 THEN CALL S
```

```
AY("#WHAT WAS THAT#")
820 IF TA<1 OR TA>3 THEN DISPLAY AT(22,1)
:"??????" :: GOTO 770
830 DISPLAY AT(22,1):" "
840 DISPLAY AT(21,19):TA :: C=C+1
850 FOR J=1 TO NBD
860 IF T(J,TD)=0 THEN 880
870 NEXT J
880 DD=J-1
890 FOR J=1 TO NBD-1
900 IF T(J,TA)=0 THEN 920
910 NEXT J
920 DA=J :: IF DA=1 THEN 950
930 IF T(DD,TD)<T(DA-1,TA)AND SS=1 THEN C
ALL SAY("YOU CAN+NOT DO THAT")
940 IF T(DD,TD)<T(DA-1,TA)THEN DISPLAY AT
(22,1):"MOUVEMENT INTERDIT" :: GOTO 690
950 K,T(DA,TA)=T(DD,TD)
960 L,T(DD,TD)=0
970 DISPLAY AT(13-DA,1+INT((K-1)/2)+9*(TA
-1))SIZE(LEN(A$(K))):A$(K)
980 DISPLAY AT(13-DD,1+INT((K-1)/2)+9*(TD
-1))SIZE(LEN(A$(K))):" "
990 IF T(1,1)=0 AND T(1,2)=0 THEN 1000 EL
SE 690
1000 GOSUB 1260
1010 DISPLAY AT(20,1):"BRAVO, VOUS AVEZ R
EUSSI EN",C;"COUPS"
1020 IF SS THEN CALL SAY("#GOOD WORK#")
1030 CM=2^NBD-1
1040 IF C=CM AND NBD<5 THEN 1110
1050 IF C=CM THEN 1130
1060 IF SS THEN CALL SAY("BUT YOU CAN PLA
Y LESS MOVE. #TRY AGAIN#")
1070 DISPLAY AT(22,1):"VOUS POUVEZ FAIRE
MIEUX","MINIMUM =";CM;"COUPS","TAPEZ PO
UR REJOUER"
1080 CALL KEY(0,K,S)
1090 IF S=0 THEN 1080
1100 GOTO 610
1110 IF SS THEN CALL SAY("VERY #GOOD WORK
#; YOU CAN+NOT DO IN LESS MOVE")
1120 DISPLAY AT(22,1):"VOUS NE POUVEZ FAI
RE MIEUX","ESSAYEZ";NBD+1;"DISQUES","TA
PEZ POUR REJOUER" :: GOTO 1080
1130 IF SS THEN CALL SAY("VERY #GOOD WORK
#; YOU FIND THE1 WAY")
1140 DISPLAY AT(22,1):"VOUS AVEZ COMPRIS_
LE TRUC","ESSAYEZ DE TROUVER LA LOI","T
APEZ POUR REJOUER" :: GOTO 1080
1150 REM OISEAU
1160 CALL MAGNIFY(3):: CALL CLEAR
1170 CALL CHAR(40,"08060301017F0300000000
000000000000080C0F2FCF80000")
1180 CALL CHAR(44,"0000007F0301030C000000
0000000000000002FCF8C0800000")
1190 CALL SPRITE(#1,40,5,50,1,0,15):: K=2
1200 FOR J=1 TO 19
1210 FOR I=1 TO 50 :: NEXT I
1220 CALL PATTERN(#1,42+K)
1230 K=-K
1240 NEXT J :: CALL DELSPRITE(#1):: CALL
CHARSET
1250 RETURN
1260 REM MUSIQUE
1270 CALL SOUND(70,262,2)
1280 CALL SOUND(70,196,2)
1290 CALL SOUND(70,262,2)
1300 CALL SOUND(70,330,2)
1310 CALL SOUND(70,392,2)
1320 CALL SOUND(70,330,2)
1330 CALL SOUND(70,262,2)
1340 CALL SOUND(70,330,2)
1350 CALL SOUND(70,349,2)
1360 CALL SOUND(70,294,2)
1370 CALL SOUND(70,247,2)
1380 CALL SOUND(70,294,2)
1390 CALL SOUND(150,262,2)
1400 CALL SOUND(150,523,2,659,2,1047,2)
1410 RETURN
1420 REM PRESENTATION
1430 CALL CLEAR
1440 DISPLAY AT(1,6):"****************"
1450 DISPLAY AT(2,6):"*TOURS DE HANOI*"
1460 DISPLAY AT(3,6):"****************"
1470 DISPLAY AT(6,5):"Vous devez faire pa
sser","de 3 a 7 disques de la 1e a","la
 3e tour."
1480 DISPLAY AT(9,5):"Vous ne pouvez plac
er","un disque sur un autre plus","peti
t."
1490 DISPLAY AT(13,5):"Essayez de reussir
 avec","un nombre minimum de coups."
1500 DISPLAY AT(17,5):"TAPEZ [1] SI VOUS
AVEZ LE","SPEECH SYNTHESIZER."
1510 CALL KEY(0,K,S)
1520 IF S=0 THEN 1510
1530 IF K=49 THEN SS=1 ELSE SS=0
1540 RETURN
```

Choisissez vos couleurs

Pascal Thomas

Vous savez sans doute que la couleur dans laquelle on programme ne peut être modifiée (noir sur cyan, ce qui n'est pas très agréable).

En tapant, en Basic Etendu et en mode immédiat, la ligne suivante avant d'écrire ou de charger un programme, vous pouvez programmer dans la couleur de votre choix :

CALL SCREEN(ECR) :: FOR I=0 TO 14 :: CALL COLOR(I,CAR,1) :: NEXT I :: RUN

Après avoir tapé "ENTER", le message :

⋆ WARNING NO PROGRAMME PRESENT ⋆

apparaît et les couleurs instituées restent à l'écran.

Toutes les commandes d'édition gardent la couleur, seule une erreur dans la syntaxe des instructions provoque la remise en place des couleurs initiales.
Il est évident qu'une fois le programme exécuté, il sera impossible de retrouver les couleurs choisies.

Trouvez le bon mot

Marianne Sutz

*N.D.L.R. : ce programme, écrit en **Basic TI**, occupe toute la mémoire vive et n'est donc pas utilisable si un contrôleur de disquettes est raccordé au TI. La classique commande "CALL FILES (1)" ne suffisant pas, les heureux possesseurs de lecteurs de disquettes devront supprimer une partie des mots et modifier les lignes 1390 et 1400 en conséquence.*

"Trouvez le bon mot" est un jeu dérivé du Mastermind. Les couleurs sont ici remplacées par des caractères utilisés pour former des noms communs de cinq lettres tirés du dictionnaire. Le programme comprend une bibliothèque de 1026 mots stockés sous forme de DATA, ce qui laisse très peu de chance de tomber deux fois de suite sur le même mot.

Utilisation du programme

Une fois le décor planté, le programme vous demande de patienter quelques instants. Ceci correspond au temps nécessaire pour le choix aléatoire d'un mot. Lorsque le TI affiche "PRET !", vous pouvez entrer votre premier essai. Le curseur, placé dans le coin supérieur gauche de l'écran, vous indique où sera affichée la lettre frappée au clavier. Vous pouvez déplacer ce curseur vers la droite en agissant sur la touche ">", et vers la gauche avec "<" (attention, il s'agit des symboles "supérieur" et "inférieur" et non des touches fléchées). La touche "0" (zéro) efface la lettre située sous le curseur, qui se déplace ensuite d'une position vers la droite. La touche "ENTER" permet la validation quelle que soit la position du curseur à condition, bien sûr, que les cinq positions soient occupées par des lettres.

Le programme autorise quinze essais, ce qui est, en règle générale, largement suffisant. Après la validation de chaque essai, le nombre de lettres bien placées sera affiché ainsi que, si nécessaire, des flèches indiquant par rapport à quel essai vous avez fait une ou plusieurs erreurs de logique. Une erreur de logique survient lorsque vous donnez une réponse impossible. Par exemple :

A C I D E - Une lettre
D E B U T - Aucune lettre

Dans ce cas, une flèche indiquant une erreur de logique sera placée à côté du mot ACIDE puisque vous aviez une lettre bien placée dans ce mot alors que, dans le mot DEBUT, aucune lettre ne correspond à une de celles constituant le premier essai.

Dans le cas où la recherche du mot vous poserait quelques problèmes, vous pouvez presser la touche "AID" (FCTN-7) et abandonner ainsi la partie.

Si le mot choisi par l'ordinateur n'est pas découvert au terme du quinzième essai, il sera automatiquement affiché et vous pourrez soit quitter le programme, soit débuter une autre partie.

■

```
100 REM  *******************
110 REM  *  Trouvez le bon  *
120 REM  *       mot        *
130 REM  *******************
140 REM  *     Basic TI     *
150 REM  *******************
160 REM
170 DATA 0000001F1F181818,000000FFFF,0000
00FFFF181818,000000F8F8181818,1818181F1
F,181818FFFF,181818F8F8
180 DATA 1818181818181818,4,4,12,12,12,12
,16,16,16,16,16,8,10,11
190 DATA tabac,barre,hampe,Kayak,masse,ve
nin,germe,idiot,index,grain,magma,ecart
,datte,rubis,arbre,jaune,recif
200 DATA mauve,toile,otite,beige,galop,pa
nne,matin,sosie,ruche,tribu,verbe,arabe
,poire,chene,satin,bille,Kyste
210 DATA hamac,malle,fugue,coude,hiver,mo
rse,voile,tafia,larme,email,poids,stade
,marge,soeur,copie,obole,reine
220 DATA joint,noyer,huche,epine,fluor,pe
nte,bulbe,valve,devot,style,guepe,niche
,repas,arene,etude,voute,bevue
230 DATA choix,stock,niece,train,forum,re
cit,Kapok,froid,delai,jouet,berge,antre
,affut,mixte,eclat,telex,soute
240 DATA autel,botte,agate,voyou,ecrin,ci
vil,debit,raton,stand,fouet,jarre,trone
,veine,diese,atoll,seuil,banne
250 DATA essai,rotor,tuyau,xenon,Kefir,as
tre,abats,harem,ultra,texte,chair,gelee
,adieu,droit,merle,balle,saint
260 DATA aieul,alize,ileon,miche,chere,ri
cin,blanc,vanne,tante,sable,usuel,digue
,noyau,foyer,patte,conte,talus
270 DATA velin,usure,swing,radis,quota,il
eum,ebene,brise,rodeo,savon,ozone,liane
,genie,drift,toast,chaud,monde
280 DATA nuage,usage,folie,purge,cosse,de
ces,jante,metal,trace,arret,acier,limon
,bulle,biere,echec,icone,serpe
290 DATA genet,valet,neveu,algue,avare,ma
ree,loque,evier,tache,salon,plant,elite
,opera,cedre,cible,zebre,urine
300 DATA panse,poule,ocean,darne,rabot,hy
ene,belon,figue,otage,avion,jeton,titre
,serie,axone,amour,canot,nurse
310 DATA gilet,loupe,hertz,album,pique,na
vet,zeste,olive,felin,acide,natte,canif
,igloo,somme,tasse,wagon,effet
320 DATA rayon,argot,rivet,orgie,bijou,go
mme,litre,dedit,etain,proie,nonne,paire
,sabot,genou,avant,adret,verve
330 DATA bysus,xylol,cerne,fable,potin,ai
gle,major,avoue,ixode,total,xyste,route
,amure,metro,butin,baton,gaffe
340 DATA slave,tanne,harpe,opium,doyen,ca
```

```
rat,fente,diode,penis,bourg,atome,veste
,rotin,etang,table,sujet,gamin
350 DATA armee,hetre,ether,motte,cause,de
vin,caget,scout,fumet,talon,orvet,pince
,debut,avril,spire,bison,glace
360 DATA macon,terre,jatte,jeune,anier,et
ape,hutte,vagin,tonne,cocon,gigot,pompe
,desir,malin,baume,sepia,quart
370 DATA actif,morue,phase,cargo,rebut,bu
gle,enjeu,sonde,jalon,yacht,douze,apnee
,filon,capot,cumul,ogive,moine
380 DATA salut,dinde,repos,ebats,jable,is
sue,bemol,rabat,sabre,oasis,lagon,geant
,decor,caban,palme,manne,azote
390 DATA weber,fruit,pavot,crane,dogue,ob
jet,mythe,tesla,arche,negre,latex,brave
,algie,hymne,salin,soupe,renne
400 DATA ictus,abime,amont,moyen,ladre,ec
ran,crime,cacao,femme,perle,joyau,divan
,ronde,herse,aloes,gorge,sauge
410 DATA amibe,neige,radar,fauve,lisse,lo
cal,colza,norme,heron,peche,biche,virus
,sapin,photo,civet,horde,flirt
420 DATA engin,coeur,appel,javel,sauge,ap
pui,tamis,fosse,alose,parot,ampli,amine
,tronc,farce,galet,queue,usine
430 DATA jabot,mache,bruit,terme,laine,de
lta,pacte,score,image,pivot,carte,grief
,dague,abord,buste,badin,firme
440 DATA cycle,hibou,bazar,atout,moyeu,si
nus,gazon,tyran,odeur,jeudi,agent,mante
,boite,bague,plomb,calme,agape
450 DATA fakir,revue,debat,chape,orgue,no
ble,prune,burin,ancre,maire,hotel,tison
,scalp,ibere,gibet,brune,basse
460 DATA ecrit,magot,cagot,rouge,pluie,fe
sse,noise,alibi,avers,palet,douve,alise
,anche,orage,garde,clown,regne
470 DATA halle,ideal,amide,exode,ruban,ta
pis,cadet,court,patin,osier,dance,autan
,beton,sirop,lacet,louve,anode
480 DATA pareo,creme,benne,verge,timon,se
cte,habit,decan,envol,aphte,prime,chant
,trait,gerbe,herbe,bande,lotus
490 DATA koala,extra,encre,bonze,bache,su
lky,haine,lotte,groom,divin,tatou,chaux
,plume,aorte,frene,nappe,vison
500 DATA clone,cygne,duree,oxyde,manie,ta
rge,ambre,lupin,lache,bonne,eleve,havre
,sceau,sucre,ruine,grace,vizir
510 DATA appat,meute,levre,ergot,coing,fo
nte,fagot,place,ovule,delit,regie,hydre
,boule,tonus,siege,cohue,lance
520 DATA akene,tombe,bahut,sante,mardi,ga
lbe,wiski,ecale,ragot,humus,neant,oisif
,egide,varan,gemme,donax,broie
530 DATA crack,cobra,caduc,houle,agres,ap
lat,atele,falot,franc,recto,selle,jarde
,alpha,flore,plage,pleur,balsa
540 DATA drome,vulve,tente,etole,arien,li
ege,ajonc,macre,gaude,ecole,mille,orobe
,chope,darse,tuile,theme,lapon
550 DATA begue,begum,benet,berce,liste,su
ite,jupon,fuite,crabe,daube,ville,beret
,bizut,aviso,biset,bicot,biais
560 DATA annee,banco,verin,heure,lange,un
ite,piece,fanon,frere,nuque,laque,acces
,arcon,gnome,valse,tapir,truie
570 DATA ombre,peste,fiche,urane,match,bi
ble,degre,ranch,zabre,salve,stele,piton
,sayon,sueur,shako,plein,yucca
580 DATA store,amble,honte,depit,mince,em
ule,urate,tempe,roman,geste,oside,catch
,livre,palan,fanal,antan,toise
590 DATA mufle,godet,eloge,diete,rhume,va
rve,imago,aigri,essor,volte,grume,tarin
,beryl,marin,poele,crise,chyme
600 DATA alese,argon,latin,ferme,fusee,ra
ble,sourd,achat,avoir,faine,perte,breme
,heros,curie,clair,depot,vague
610 DATA uvule,lutte,obese,messe,groin,et
ron,marne,cabus,colle,diner,torse,tanin
,logis,sicle,ajout,arome,borne
620 DATA laser,pacha,polka,fille,furie,jo
ule,large,ajour,avent,grade,verre,terne
,cabas,corne,epure,ouate,plate
630 DATA breve,berne,devis,regal,heurt,se
rge,punch,pelle,signe,saule,sprat,potee
,bilan,bigle,final,argus,accul
640 DATA alios,joncs,tripe,recul,golfe,ni
tre,cabot,cache,puree,forme,abces,agree
,temps,butte,stere,juste,greve
650 DATA harde,vesse,utile,egard,abaca,gl
ane,datif,tarse,balai,macle,poete,corde
,canne,hotte,abois,alene,lourd
660 DATA adent,salol,strie,alpax,coppa,pr
ose,palot,bigot,huppe,culex,capre,vigne
,taupe,auget,yeuse,laize,round
670 DATA rafle,murex,armet,carex,crete,du
che,these,lapin,sacre,sphex,amant,azyme
,buron,agnat,zeine,lopin,paume
680 DATA nabab,joute,lunch,aimee,genre,gu
ppy,vouge,tomme,cahot,cuvee,eveil,passe
,faste,frise,hevea,sioux,poste
690 DATA semis,aleph,prude,butee,banjo,fr
ont,aspic,homme,demon,metre,envoi,jetee
,tiers,riche,gifle,orque,caste
700 DATA coche,corps,axile,armon,allee,ab
out,bravo,bombe,bidet,bagou,staff,sport
,spore,seche,gouge,grand,gibus
710 DATA envie,mucus,mulet,matou,melon,de
tte,deuil,delco,rente,renom,rebus,haire
,huile,poing,point,pouce,poupe
720 DATA filet,tenia,trois,ordre,nasse,no
tus,yacca,xerus,wharf,vesce,vanda,freon
,buvee,cagne,copte,corso,couac
730 DATA egout,isart,frais,grive,atlas,bo
sse,goule,pagne,union,grele,ongle,cabri
```

```
,ahuri,lecon,magie,brume,asile
740 DATA futur,radio,quete,chute,fusil,fl
air,bocal,oncle,athee,beche,fenil,cable
,coque,gypse,bidon,carme,idole
750 DATA lycee,gaine,boeuf,lampe,faune,va
che,guide,galon,doigt,jambe,bride,xeres
,blond,cidre,tigre,epice,seize
760 DATA tenor,angle,hache,micro,tango,om
ega,saxon,silex,dalle,zloty,faute,nacre
,adage,poker,furet,motel,paria
770 DATA cirre,coton,scion,arete,augee,at
oca,carpe,ecume,buche,appas,baril,serum
,whist,aulne,totem,ecrou,volve
780 DATA hypne,flash,megot,doute,maser,es
ter,degat,label,fiole,ozene,myrte,purin
,foret,patio,stage,cadre,carde
790 DATA epieu,enfer,pomme,fumee,frein,to
nka
800 CALL CLEAR
810 CALL SCREEN(5)
820 OPTION BASE 1
830 DIM E$(15,5),L$(5),E(15)
840 FOR B=152 TO 159
850 READ B$
860 CALL CHAR(B,B$)
870 NEXT B
880 CALL CHAR(136,"008282FEFE")
890 CALL CHAR(144,"0010307FFE7F301")
900 FOR B=3 TO 16
910 READ C
920 CALL COLOR(B,C,1)
930 NEXT B
940 CALL HCHAR(1,3,153,29)
950 CALL HCHAR(18,3,153,29)
960 CALL HCHAR(19,3,153,29)
970 CALL HCHAR(24,3,153,29)
980 CALL VCHAR(2,2,159,22)
990 CALL VCHAR(2,12,159,16)
1000 CALL VCHAR(2,14,159,16)
1010 CALL VCHAR(2,29,159,16)
1020 CALL VCHAR(2,32,159,22)
1030 CALL HCHAR(1,2,152)
1040 CALL HCHAR(19,2,152)
1050 CALL HCHAR(1,12,154)
1060 CALL HCHAR(1,14,154)
1070 CALL HCHAR(1,29,154)
1080 CALL HCHAR(1,32,155)
1090 CALL HCHAR(19,32,155)
1100 CALL HCHAR(18,2,156)
1110 CALL HCHAR(24,2,156)
1120 CALL HCHAR(18,12,157)
1130 CALL HCHAR(18,14,157)
1140 CALL HCHAR(18,29,157)
1150 CALL HCHAR(18,32,158)
1160 CALL HCHAR(24,32,158)
1170 C=30
1180 FOR B=1 TO 15
1190 L=B+1
1200 IF B<10 THEN 1230
1210 T$=STR$(B)
1220 GOTO 1240
1230 T$=" "&STR$(B)
1240 GOSUB 2870
1250 NEXT B
1260 T$="- TROUVEZ LE BON MOT -"
1270 L=20
1280 C=7
1290 GOSUB 2870
1300 T$="VEUILLEZ PATIENTER QUELQUES"
1310 L=22
1320 C=4
1330 GOSUB 2870
1340 T$="INSTANTS SVP"
1350 L=23
1360 C=12
1370 GOSUB 2870
1380 RANDOMIZE
1390 N=INT(RND*1026)+1
1400 FOR B=1 TO 1026
1410 READ B$
1420 IF B<>N THEN 1440
1430 M$=B$
1440 NEXT B
1450 FOR B=1 TO 5
1460 L$(B)=SEG$(M$,B,1)
1470 NEXT B
1480 CALL HCHAR(3,3,136)
1490 FOR B=20 TO 23
1500 CALL HCHAR(B,3,32,29)
1510 NEXT B
1520 T$="PRET !"
1530 L=20
1540 C=15
1550 GOSUB 2870
1560 CALL SOUND(5,880,0)
1570 CALL SOUND(5,988,0)
1580 CALL SOUND(5,1047,0)
1590 CX=1
1600 LX=LX+1
1610 CR=0
1620 CALL KEY(0,K,S)
1630 IF S<1 THEN 1620
1640 IF K=1 THEN 2570
1650 IF K=13 THEN 1920
1660 IF K<>48 THEN 1700
1670 E$(LX,CX)=""
1680 CALL HCHAR(LX+1,CX*2+1,32)
1690 GOTO 1860
1700 IF K=44 THEN 1780
1710 IF K=46 THEN 1830
1720 IF (K<65)+(K>90)*(K<97)+(K>122)THEN
1620
1730 IF K>90 THEN 1750
1740 K=K+32
1750 CALL HCHAR(LX+1,CX*2+1,K)
1760 E$(LX,CX)=CHR$(K)
1770 GOTO 1830
1780 IF CX=1 THEN 1620
```

```
1790 CALL HCHAR(LX+2,CX*2+1,32)
1800 CX=CX-1
1810 CALL HCHAR(LX+2,CX*2+1,136)
1820 GOTO 1620
1830 IF CR THEN 1860
1840 CALL VCHAR(2,13,32,15)
1850 CALL HCHAR(20,8,32,20)
1860 CR=CR+1
1870 IF CX=5 THEN 1620
1880 CALL HCHAR(LX+2,CX*2+1,32)
1890 CX=CX+1
1900 CALL HCHAR(LX+2,CX*2+1,136)
1910 GOTO 1620
1920 DR=0
1930 FOR B=1 TO 5
1940 IF E$(LX,B)<>"" THEN 1960
1950 DR=1
1960 NEXT B
1970 IF DR THEN 1620
1980 FOR B=1 TO 5
1990 IF E$(LX,B)<>L$(B)THEN 2010
2000 E(LX)=E(LX)+1
2010 NEXT B
2020 ON E(LX)+1 GOTO 2030,2050,2070,2090,
2110,2130
2030 T1$="aucune"
2040 GOTO 2140
2050 T1$="   une"
2060 GOTO 2140
2070 T1$="  deux"
2080 GOTO 2140
2090 T1$=" trois"
2100 GOTO 2140
2110 T1$="quatre"
2120 GOTO 2140
2130 T1$="  cinq"
2140 L=LX+1
2150 C=15
2160 IF E(LX)>1 THEN 2190
2170 T$=T1$&" lettre"
2180 GOTO 2200
2190 T$=T1$&" lettres"
2200 GOSUB 2870
2210 IF LX=1 THEN 2430
2220 DR=0
2230 FOR B=1 TO LX-1
2240 V=0
2250 FOR B2=1 TO 5
2260 IF E$(LX,B2)<>E$(B,B2)THEN 2280
2270 V=V+1
2280 NEXT B2
2290 IF V=E(B)THEN 2330
2300 CALL SOUND(10,1000,0)
2310 CALL HCHAR(B+1,13,144)
2320 DR=DR+1
2330 NEXT B
2340 IF DR THEN 2360
2350 GOTO 2430
2360 IF DR>1 THEN 2390
2370 T$=CHR$(144)&" ERREUR DE LOGIQUE"
2380 GOTO 2400
2390 T$=CHR$(144)&" ERREURS DE LOGIQUE"
2400 L=20
2410 C=8
2420 GOSUB 2870
2430 IF E(B)<5 THEN 2490
2440 T$="LE MOT ETAIT BIEN : "&M$
2450 L=21
2460 C=5
2470 GOSUB 2870
2480 GOTO 2650
2490 IF LX<>14 THEN 2560
2500 CALL COLOR(3,9,1)
2510 CALL COLOR(4,9,1)
2520 T$="DERNIER ESSAI !"
2530 L=21
2540 C=10
2550 GOSUB 2870
2560 IF LX<>15 THEN 2620
2570 T$="LE MOT ETAIT : "&M$
2580 L=21
2590 C=7
2600 GOSUB 2870
2610 GOTO 2650
2620 CALL HCHAR(LX+2,3,32,9)
2630 CALL HCHAR(LX+3,3,136)
2640 GOTO 1590
2650 T$="UNE AUTRE PARTIE ?"
2660 C=9
2670 L=22
2680 GOSUB 2870
2690 T$="(O/N)"
2700 C=15
2710 L=23
2720 GOSUB 2870
2730 CALL KEY(0,K,S)
2740 IF S=0 THEN 2730
2750 IF (K=78)+(K=110)THEN 2930
2760 IF (K<>79)*(K<>111)THEN 2730
2770 CALL CLEAR
2780 FOR B=1 TO 15
2790 E(B)=0
2800 FOR B2=1 TO 5
2810 E$(B,B2)=""
2820 NEXT B2
2830 NEXT B
2840 RESTORE
2850 LX=0
2860 GOTO 840
2870 PC=1
2880 FOR X=C TO C+LEN(T$)-1
2890 CALL HCHAR(L,X,ASC(SEG$(T$,PC,1)))
2900 PC=PC+1
2910 NEXT X
2920 RETURN
2930 CALL CLEAR
2940 END
```

Jeu de la vie

Gerbert Cinquet

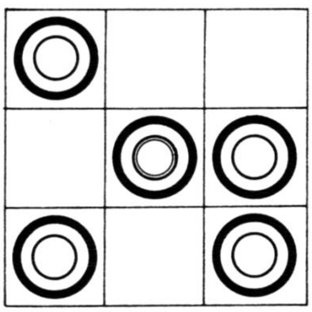
Le jeu de la vie, imaginé par le mathématicien Conway, est un programme qui a été adapté à la plupart des ordinateurs; il fait donc partie des "classiques". Il s'agit d'un jeu passif puisque l'utilisateur ne peut pas modifier le déroulement du programme.

Nous avons fait le choix d'écrire deux programmes effectuant exactement le même travail; le premier est écrit en **Basic TI** alors que le second est réalisé en **Assembleur** et est prévu pour fonctionner avec seulement la *Mini-mémoire*.

L'espace mémoire utilisé par le programme écrit en Basic TI étant important, il est impossible de faire tourner le *jeu de la vie* si un lecteur de disquettes est raccordé au TI, même en employant *CALL FILES (1)*.

L'écran de l'ordinateur visualise une population de cellules qui peuvent mourir , survivre ou naître.

Chaque cellule peut avoir jusqu'à 8 voisines :



sauf celles qui sont situées sur un côté de l'écran (5 voisines) ou dans un angle (3).

Dans un premier temps, le programme affiche aléatoirement la génération 0. Le nombre de cellules peut varier entre 255 et 510. Ensuite, le calcul de la génération suivante est réalisé, et il suffit d'appuyer sur la touche *espace* pour visualiser la nouvelle génération. Enfin, une pression sur la touche *N* permet l'initialisation d'une nouvelle génération 0. Voyons quelles sont les règles qui determinent l'avenir de nos cellules :

- 0 ou 1 cellule adjacente : mort par isolement (la cellule est blanche);
- 2 cellules adjacentes : survie (la cellule est rouge);
- 3 cellules adjacentes : naissance si la position considérée est vide (la cellule est bleue);
- plus de 4 cellules adjacentes : mort par surpopulation (cellule blanche).

Si le programme écrit en Basic se passe de commentaires, il n'en va pas de même pour la version écrite en Assembleur, surtout si vous débutez dans cette discipline; quelques explications ne seront peut-être pas superflues. Il va de soi que si l'assembleur n'a plus de secrets pour vous, vous pouvez passer allègrement la suite de cet article.

Version "Assembleur"

Avant d'aller plus loin, nous vous conseillons de charger le programme en mémoire car, en voyant fonctionner le *jeu de la vie*, vous pourrez comprendre plus aisément les explications qui vont suivre.

Si vous disposez de la cassette d'accompagnement de ce numéro, aucun problème, puisque vous n'avez qu'à charger le programme (avec l'option *L* de *EASY BUG*) et à le faire démarrer (avec l'option *RUN* du menu principal de la *Mini-mémoire*, le nom du programme étant "JEUVIE").

Sinon, il va falloir assembler le programme. Pour cela, il faut bien sûr charger l'Assembleur ligne par ligne et lancer son exécution. Vous pouvez alors commencer à entrer le programme.

Attention : les trois premières colonnes du listing représentent les numéros de ligne, les adresses en mémoire et le contenu de ces mêmes adresses; il ne faut donc taper que ce qui suit ces trois colonnes.

Vous pouvez remarquer que le code source ne comporte pas d'étiquettes; cela est dû au fait que l'espace mémoire disponible pour recevoir le code objet, lorsque l'assembleur est en mémoire, est des plus réduit, et que les étiquettes utilisent une partie de cette zone. Pour que le programme puisse être assemblé avec l'assembleur ligne par ligne, il fallait donc que les branchements et sauts soient notifiés par des adresses réelles.
Le code source ne comportant pas d'information dans la zone des étiquettes, chaque entrée de ligne pourra débuter par une pression sur la barre d'espacement, ce qui a pour effet de placer le curseur au début de la zone des codes mnémoniques.

Il faut enfin signaler qu'il n'est pas nécessaire de saisir les lignes commençant par un astérisque, puisqu'elles sont uniquement destinées à recevoir des remarques, et ne sont pas prises en compte par l'assembleur..

Nous pouvons maintenant passer aux explications du programme.

Lorsqu'on débute en informatique, la première chose que l'on essaye de faire, c'est afficher un mot, une phrase ou un nombre sur l'écran.

En Basic, même si l'on n'a jamais touché à un ordinateur, cela ne pose généralement pas de problème; on se rend très vite compte que pour afficher *TOTO*, il faut faire *PRINT TOTO - RETURN*.

En assembleur, le problème est tout autre, et le caractère non conversationnel de ce langage ne facilite pas les choses; on ne peut pas avoir de résultat immédiat, ce qui fait que les essais "par tâtonnement", très faciles à réaliser en Basic, sont ici impossibles.

Le programme que nous vous proposons étant essentiellement basé sur l'affichage, il va nous permettre d'éclaircir certains points.

Organisation de la mémoire vidéo

En assembleur, les *PRINT, CALL HCHAR, CALL COLOR, etc...* n'existent pas; il faut donc les simuler, sans pour autant essayer d'imiter le Basic (à langages différents, méthodes différentes).
Pour cela, il est indispensable de savoir comment est organisée la mémoire vive du processeur vidéo, utilisée pour la gestion de l'écran.

Dans cet article, nous nous contenterons d'examiner le mode graphique, qui est le seul qui soit utilisable directement par le Basic TI. Il faut toutefois rappeler que le TI-99/4A peut employer quatre modes d'affichage :

- mode graphique : 24 lignes de 32 caractères, soit 768 caractères;
- mode texte : 24 lignes de 40 caractères, soit 960 caractères;
- mode multicolore : 48 lignes de 64 petits pavés de 4*4 pixels (il est possible d'indiquer la couleur de chaque pavé mais ils ne sont pas redéfinissables; ce mode ne présente donc pas un grand intérêt);
- mode "Bit-Map" : les 49152 pixels constituant l'affichage peuvent être gérés séparément, et il est possible d'indiquer les couleurs des points "off" et "on" pour chaque ligne horizontale de huit pixels; il s'agit du mode le plus intéressant sur le TI.

En assembleur, et en mode graphique, nous disposons de trois tables

situées en mémoire vidéo.

Table d'écran

Cette table est une copie en temps réel de ce qui est affiché sur l'écran. Elle contient les codes ASCII des caractères et utilise 768 octets (puisque l'on peut afficher 768 caractères en mode graphique). Pour simuler le *CALL CLEAR* du Basic TI, il suffit de placer la valeur >20 (32 en décimal; soit le code ASCII du caractère "espace") dans les 768 octets constituant la table d'écran.

Dans notre programme, cette table commence à l'adresse >0000 et s'étend jusqu'à l'adresse >02FF (767), ce qui nous fait bien 768 octets. >0000 constitue l'adresse de base de la table qui, nous le verrons un peu plus loin, peut très bien être située à un autre endroit.

Si l'adresse de base de la table d'écran est >0000, les adresses >0000 à >001F (31) contiennent les codes ASCII des caractères affichés sur la première ligne de l'écran (en haut, et de gauche à droite), les adresses >0020 (32) à >003D (61) contiennent les codes des caractères affichés sur la deuxième ligne, et ainsi de suite jusqu'à >02FF (767) qui contient le code du caractère placé dans le coin inférieur droit de l'écran.
Ainsi, si l'on place la valeur >41 (65) dans l'adresse >02FF, nous obtenons l'affichage d'un "A" dans le coin inférieur droit de l'écran (>41 est le code ASCII de la lettre "A").

Pour placer le caractère *X* à l'intersection de la ligne *L* et la colonne *C*, nous pouvons utiliser la formule suivante :

A + (32 ⋆ L + C - 33)

dans laquelle *A* est l'adresse de base de notre table (>0000 dans notre programme). Le résultat de ce calcul est l'adresse qui correspondrait, en TI Basic à *CALL CHAR (L,C,X)*.

Le moyen le plus simple pour afficher un caractère est d'utiliser l'utilitaire *VSBW*, situé physiquement à l'adresse >6024 (dans la ROM de la Mini-mémoire). Deux paramètres sont requis :

- l'adresse dans la table image écran, qui doit être contenue par le registre 0;
- le code ASCII du caractère à afficher, qui doit être situé dans l'octet de poids fort (octet de gauche) du registre 1.

Concrètement, pour afficher la lettre "A" dans le coin supérieur droit de l'écran, on peut utiliser :

```
LI R0,>001F
LI R1,>4100
BLWP @>6024
```

Comme nous le laissions entrevoir précédemment, il est possible de modifier l'adresse de base de la table. Pour cela, il faut charger une valeur dans le registre 2 du processeur vidéo.
Cette valeur sera multipliée par >400 (1024) par le processeur, ce qui fait que les adresses de base possibles sont :

valeur 0 : >0000
valeur 1 : >0400 (1024)
valeur 2 : >0800 (2048)
valeur 3 : >0C00 (3072)
et ainsi de suite jusqu'à :
valeur 15 : >3C00 (15360)

Pour modifier le contenu d'un registre vidéo, on peut utiliser la routine VWTR, située à l'adresse >6034, et qui requiert le numéro du registre à modifier dans l'octet de poids fort du registre 0, et la valeur à transférer dans l'octet de poids faible du même registre. Si l'on veut que l'adresse de base de la table d'écran soit égale à >0400, deux lignes suffisent :

```
LI RO,>0201
BLWP @>6034
```

Table des descripteurs

Cette table, appelée parfois table des patrons, contient la définition de la forme des caractères. Elle est utilisée, en Basic TI, par l'instruction *CALL CHAR* mais, en Assembleur, elle est beaucoup plus étendue puisqu'elle contient la définition de 256 caractères.

Tout comme en Basic, une partie de cette table contient des caractères prédéfinis par le programme moniteur; ce sont d'ailleurs les mêmes.

Dans *le jeu de la vie*, l'adresse de base de la table des descripteurs est >0800 (2048). Puisque l'on peut définir ou redéfinir 256 caractères, et que chaque caractère utilise 8 octets (comme en Basic), la table utilise 256⋆8 soit 2048 octets ou 2 Ko.

Les octets situés aux adresses >0800 à >0807 contiennent la définition du caractère 0; ceux situés aux adresses >0808 à >080F définissent le caractère 1, et ainsi de suite jusqu'aux adresses >0FF8 à >0FFF, utilisées pour la définition du caractère >FF (255).

Pour calculer les adresses à modifier afin de redéfinir le caractère X (X représente le numéro du caractère et non la lettre *X*), on peut employer la formule :

A + X ⋆ 8

dans laquelle *A* représente l'adresse de base de la table. Le résultat de l'opération nous indique le premier des 8 octets à modifier. Par exemple, pour redéfinir le caractère >41 (65) :

2048 + 65 ⋆ 8 = 2568 ou >0A08

Il faut donc modifier le contenu des adresses >0A08 à >0A0F.

La routine VMBW, située en >6028 est très pratique lorsqu'il faut transférer plusieurs octets en mémoire vidéo (elle est d'ailleurs faite pour cela !). Cette routine requiert 3 paramètres :

- dans le registre 0 : l'adresse en mémoire vidéo où sera placé le premier octet transféré;
- dans le registre 1 : l'adresse du tampon de données en mémoire vive (directive *DATA*);
- dans le registre 2 : le nombre d'octets à transférer (8 pour définir un caractère).

Si nous voulons que le caractère 0 ait la forme d'un carré, et en supposant que le tampon se situe à l'adresse >7800, on peut écrire :

```
DATA >FF81
DATA >8181
DATA >8181
DATA >81FF
LI R0,>0800
LI R1,>7800
LI R2,>0008
BLWP @>6028
```

L'adresse de base de la table des descripteurs peut aussi être modifiée. Pour cela, il faut cette fois charger une valeur dans le registre 4 du processeur vidéo. Les possibilités sont plus limitées que pour la table d'écran car la valeur n'est pas multipliée par >400, mais par >800 (2048). Vous avez certainement compris que, dans notre programme, le registre vidéo 4 contient 1, puisque l'adresse de base de la table est 2048.

La table des couleurs

Comme nous venons de le voir, nous disposons, en Assembleur, d'un jeu de 256 caractères. Comme en Basic TI, la couleur des points "on" (allumés) et celle des points "off" (éteints) peut être modifiée, et ce par groupes de 8 caractères, ce qui nous fait donc 256 / 8 = 32 groupes.

Chaque groupe de caractères utilise un octet divisé en deux quartets (4 bits) :

- quartet de poids fort (le plus à gauche) : couleur des points "on" du groupe considéré;
- quartet de poids faible (le plus à droite) : couleur des points "off".

La table des couleurs occupe donc 32 octets et débute, dans le programme proposé, à l'adresse >0380. Ainsi, l'adresse >0380 (896) contient les couleurs des caractères 0 à 7, >0381 contient les couleurs des caractères 8 à 15, etc...

En binaire et sur 4 bits, on ne peut représenter que des valeurs comprises entre 0000 (0) et 1111 (15); les

codes de couleur ne sont donc pas les valeurs 1 à 16 comme en Basic mais 0 à 15, soit la couleur en Basic - 1 : *incolore* = 0 et *blanc* = 15.

Pour donner les couleurs *blanc sur noir* aux caractères faisant partie du groupe 0 (0 à 7), on peut utiliser la même méthode que pour afficher un caractère sur l'écran :

LI R0,>0380
LI R1,>F200
BLWP @>6024

Il est aussi possible de modifier l'emplacement de la table, en chargeant une valeur dans le registre 3 du processeur vidéo. Cette valeur est multipliée par >40 (64); il y a donc un grand nombre d'emplacements possibles.

Dans notre exemple, l'adresse de base de la table des couleurs étant >0380 (896), le registre vidéo 3 contient la valeur 896 / 64 = 14 (E), qui correspond à la valeur par défaut avec l'Editeur/Assembleur ou la Mini-mémoire.

Couleur de l'écran

Pour remplacer l'instruction *CALL SCREEN* du Basic TI, il faut employer le registre 7 du processeur vidéo.

En mode graphique, seul le quartet de poids faible de ce registre est utilisé, le quartet de poids fort étant destiné à recevoir la couleur des caractères en mode "texte".

La routine *VWTR*, que nous avons utilisée précédemment pour modifier l'adresse de base des tables, peut remplacer le *CALL SCREEN* :

LI RO,>0702
BLWP @>6034

On obtient alors un écran noir.

Temps d'exécution

Si les résultat obtenus sont les mêmes avec le programme Basic qu'avec celui écrit en Assembleur, les temps nécessaires au calcul d'une nouvelle génération diffèrent sensiblement.

En Basic TI :

Calcul d'une génération : 7 minutes et 49 secondes;
Affichage d'une génération : 45 secondes.

En Assembleur :

Calcul d'une génération : environ 0,5 seconde;
Affichage d'une génération : certainement inférieur à 1/50ème de seconde.

Autrement dit, pour parvenir à la 100ème génération, il faut, en Basic, 14 heures, 16 minutes et 40 secondes, alors qu'en Assembleur 50 secondes suffisent !

Un autre point non négligeable : le programme écrit en Basic occupe 13864 octets, alors que la version en Assembleur occupe seulement 2156 octets. ■

```
100 REM ********************
110 REM *  Jeu de la vie   *
120 REM ********************
130 REM *     Basic TI     *
140 REM ********************
150 REM
160 REM initialisations
170 REM
180 CALL CLEAR
190 OPTION BASE 1
200 DIM T1(22,30),T2(22,30),TL(8),TC(8)
210 DATA -1,-1,-1,0,-1,1,0,1,1,1,1,0,1,-1
,0,-1
220 FOR B=1 TO 8
230 READ L,C
240 TL(B)=L
250 TC(B)=C
260 NEXT B
270 C$="003C665A5A663C"
280 REM
290 REM couleurs et redefinitions
300 REM
310 FOR B=136 TO 152 STEP 8
320 CALL CHAR(B,C$)
330 NEXT B
340 CALL CHAR(63,"0")
350 FOR B=3 TO 7
360 CALL COLOR(B,16,13)
370 NEXT B
380 CALL COLOR(1,11,11)
390 CALL COLOR(14,7,11)
400 CALL COLOR(15,6,11)
410 CALL COLOR(16,16,11)
420 CALL SCREEN(13)
430 CALL CLEAR
440 REM
450 REM affiche bordures
460 REM
470 CALL HCHAR(1,1,63,32)
480 CALL HCHAR(24,1,63,32)
490 CALL VCHAR(2,1,63,22)
500 CALL VCHAR(2,32,63,22)
510 REM
520 REM premiere generation
530 REM
540 G=0
550 FOR B=1 TO 22
560 FOR B2=1 TO 30
570 T1(B,B2)=32
580 NEXT B2
590 NEXT B
600 REM
610 REM nombre de cellules
620 REM
630 RANDOMIZE
640 B1=INT(RND*256)+255
650 FOR B=1 TO B1
660 REM
670 REM repartition cellules
680 REM
690 RANDOMIZE
700 L=INT(RND*22)+1
710 C=INT(RND*30)+1
720 IF T1(L,C)=136 THEN 690
730 T1(L,C)=136
740 CALL HCHAR(L+1,C+1,136)
750 NEXT B
760 L=24
770 C=2
780 T$="GENERATION:000????CELLULES:"
790 GOSUB 1550
800 T$=STR$(B1)
```

```
810 C=29
820 GOSUB 1460
830 CALL SOUND(1,1000,0)
840 REM
850 REM calcul generation n+1
860 REM
870 G=G+1
880 CT=0
890 FOR L=1 TO 22
900 FOR C=1 TO 30
910 CR=0
920 FOR B=1 TO 8
930 LX=L+TL(B)
940 IF (LX<1)+(LX>22)THEN 990
950 CX=C+TC(B)
960 IF (CX<1)+(CX>30)THEN 990
970 IF (T1(LX,CX)<>136)*(T1(LX,CX)<>144)T
HEN 990
980 CR=CR+1
990 NEXT B
1000 IF (T1(L,C)=136)+(T1(L,C)=144)THEN 1
070
1010 IF CR=3 THEN 1040
1020 T2(L,C)=32
1030 GOTO 1130
1040 T2(L,C)=144
1050 CT=CT+1
1060 GOTO 1130
1070 IF (CR<=1)+(CR>3)THEN 1120
1080 IF CR>3 THEN 32767
1090 T2(L,C)=136
1100 CT=CT+1
1110 GOTO 1130
1120 T2(L,C)=152
1130 NEXT C
1140 NEXT L
1150 REM
1160 REM transfert tableau T2->T1
1170 REM
1180 FOR L=1 TO 22
1190 FOR C=1 TO 30
1200 T1(L,C)=T2(L,C)
1210 NEXT C
1220 NEXT L
1230 CALL SOUND(1,1000,0)
1240 CALL KEY(0,K,S)
1250 IF K=78 THEN 430
1260 IF K<>32 THEN 1240
1270 REM
1280 REM transfert tableau T2->affichage
1290 REM
1300 FOR L=1 TO 22
1310 FOR C=1 TO 30
1320 CALL HCHAR(L+1,C+1,T2(L,C))
1330 NEXT C
1340 NEXT L
1350 C=29
1360 L=24
1370 T$=STR$(CT)
1380 GOSUB 1460
1390 C=13
1400 T$=STR$(G)
1410 GOSUB 1460
1420 GOTO 870
1430 REM
1440 REM "display at using"
1450 REM
1460 IF LEN(T$)=3 THEN 1550
1470 IF LEN(T$)=2 THEN 1500
1480 T$="00"&T$
1490 GOTO 1550
1500 T$="0"&T$
1510 GOTO 1550
1520 REM
1530 REM "display at"
1540 REM
1550 X=1
1560 FOR B=C TO C+LEN(T$)-1
1570 CALL HCHAR(L,B,ASC(SEG$(T$,X,1)))
1580 X=X+1
1590 NEXT B
1600 RETURN
```

```
0001 774A        AORG >774A
0002 774A        BSS  704
0003 7A0A        BSS  768
0004 7D0A        BSS  2
0005 7D0C        BSS  2
0006 7D0E        BSS  2
0007 7D10        BSS  32
0008 7D30 0000   DATA 0,0,0,0
     7D32 0000
     7D34 0000
     7D36 0000
0009 7D38 003C   DATA >3C,>665A,>5A66,>3C00
     7D3A 665A
     7D3C 5A66
     7D3E 3C00
0010 7D40 6A4A   DATA >6A4A,>EA00
     7D42 EA00
0011 7D44 0108   DATA >108,>1000,>2000
     7D46 1000
     7D48 2000
0012 7D4A FF20   DATA >FF20,>4E00
     7D4C 4E00
0013 7D4E FFDF   DATA -33,1,1,32,32,-1,-1,-32
     7D50 0001
     7D52 0001
     7D54 0020
     7D56 0020
     7D58 FFFF
     7D5A FFFF
     7D5C FFE0
0014 7D5E 0F00   DATA >F00
0015 7D60 03E8   DATA 1000
0016 7D62 0064   DATA 100
0017 7D64 000A   DATA 10
0018 7D66   47   TEXT 'GENERATION:000    CELLULES: '
0019            * charge pointeur ws
0020 7D82 02E0   LWPI >7D10
     7D84 7D10
0021            * couleurs et redefinitions
0022 7D86 0200   LI   R0,>7CC
```

```
     7D88 07CC
0023 7D8A 0420   BLWP @>6034
     7D8C 6034
0024 7D8E 0200   LI   R0,>800
     7D90 0800
0025 7D92 0201   LI   R1,>7D30
     7D94 7D30
0026 7D96 0202   LI   R2,16
     7D98 0010
0027 7D9A 0420   BLWP @>6028
     7D9C 6028
0028 7D9E 0200   LI   R0,>840
     7DA0 0840
0029 7DA2 0201   LI   R1,>7D38
     7DA4 7D38
0030 7DA6 0202   LI   R2,8
     7DA8 0008
0031 7DAA 0420   BLWP @>6028
     7DAC 6028
0032 7DAE 0200   LI   R0,>880
     7DB0 0880
0033 7DB2 0420   BLWP @>6028
     7DB4 6028
0034 7DB6 0200   LI   R0,>380
     7DB8 0380
0035 7DBA 0201   LI   R1,>7D40
     7DBC 7D40
0036 7DBE 0202   LI   R2,3
     7DC0 0003
0037 7DC2 0420   BLWP @>6028
     7DC4 6028
0038 7DC6 0200   LI   R0,>384
     7DC8 0384
0039 7DCA 0201   LI   R1,>FC00
     7DCC FC00
0040 7DCE 0420   BLWP @>6024
     7DD0 6024
0041 7DD2 0580   INC  R0
0042 7DD4 0280   CI   R0,>38B
     7DD6 038B
0043 7DD8 16FA   JNE  >7DCE
0044             * premiere generation
0045 7DDA 04E0   CLR  @>7D0C
     7DDC 7D0C
0046 7DDE 04C1   CLR  R1
0047 7DE0 0200   LI   R0,>7A2A
     7DE2 7A2A
0048 7DE4 CC01   MOV  R1,*R0+
0049 7DE6 0280   CI   R0,>7CEA
     7DE8 7CEA
0050 7DEA 16FC   JNE  >7DE4
0051             * affiche bordures
0052 7DEC 0200   LI   R0,>7A0A
     7DEE 7A0A
0053 7DF0 0201   LI   R1,>2000
     7DF2 2000
0054 7DF4 DC01   MOVB R1,*R0+
0055 7DF6 0280   CI   R0,>7A2A
     7DF8 7A2A
0056 7DFA 16FC   JNE  >7DF4
0057 7DFC D401   MOVB R1,*R0
0058 7DFE 0220   AI   R0,32
     7E00 0020
0059 7E02 0280   CI   R0,>7CEA
     7E04 7CEA
0060 7E06 16FA   JNE  >7DFC
0061 7E08 DC01   MOVB R1,*R0+
0062 7E0A 0280   CI   R0,>7D09
     7E0C 7D09
0063 7E0E 16FC   JNE  >7E08
0064 7E10 D401   MOVB R1,*R0
```

```
0065 7E12 0220   AI   R0,-32
     7E14 FFE0
0066 7E16 0280   CI   R0,>7A29
     7E18 7A29
0067 7E1A 16FA   JNE  >7E10
0068             * affiche texte
0069 7E1C 0200   LI   R0,>7CEB
     7E1E 7CEB
0070 7E20 0201   LI   R1,>7D66
     7E22 7D66
0071 7E24 DC31   MOVB *R1+,*R0+
0072 7E26 0281   CI   R1,>7D82
     7E28 7D82
0073 7E2A 16FC   JNE  >7E24
0074             * nombre de cellules
0075 7E2C 06A0   BL   @>7F66
     7E2E 7F66
0076 7E30 C120   MOV  @>83C0,R4
     7E32 83C0
0077 7E34 0984   SRL  R4,8
0078 7E36 0224   AI   R4,>FF
     7E38 00FF
0079 7E3A C804   MOV  R4,@>7D0E
     7E3C 7D0E
0080             * repartition cellules
0081 7E3E C1C4   MOV  R4,R7
0082 7E40 06A0   BL   @>7F66
     7E42 7F66
0083 7E44 C3E0   MOV  @>83C0,R15
     7E46 83C0
0084 7E48 096F   SRL  R15,6
0085 7E4A 028F   CI   R15,734
     7E4C 02DE
0086 7E4E 15F8   JGT  >7E40
0087 7E50 DBEF   MOVB @>7A0A(R15),@>7A0A(R15)
     7E52 7A0A
     7E54 7A0A
0088 7E56 16F4   JNE  >7E40
0089 7E58 DBE0   MOVB @>7D44,@>7A0A(R15)
     7E5A 7D44
     7E5C 7A0A
0090 7E5E 0607   DEC  R7
0091 7E60 16EF   JNE  >7E40
0092 7E62 04C0   CLR  R0
0093 7E64 0201   LI   R1,>7A0A
     7E66 7A0A
0094 7E68 0202   LI   R2,768
     7E6A 0300
0095 7E6C 0420   BLWP @>6028
     7E6E 6028
0096             * affiche nombre cellules
0097 7E70 C160   MOV  @>7D0E,R5
     7E72 7D0E
0098 7E74 06A0   BL   @>7F78
     7E76 7F78
0099 7E78 02FC   DATA >02FC
0100             * calcul generation n+1
0101 7E7A 0200   LI   R0,>7A2A
     7E7C 7A2A
0102 7E7E 0206   LI   R6,>774A
     7E80 774A
0103 7E82 04E0   CLR  @>7D0E
     7E84 7D0E
0104 7E86 9810   CB   *R0,@>7D48
     7E88 7D48
0105 7E8A 1602   JNE  >7E90
0106 7E8C DDB0   MOVB *R0+,*R6+
0107 7E8E 102D   JMP  >7EEA
0108 7E90 D210   MOVB *R0,R8
0109 7E92 04C3   CLR  R3
0110 7E94 04C7   CLR  R7
```

```
0111 7E96 A023   A    @>7D4E(R3),R0
     7E98 7D4E
0112 7E9A 9810   CB   *R0,@>7D44
     7E9C 7D44
0113 7E9E 1303   JEQ  >7EA6
0114 7EA0 9810   CB   *R0,@>7D45
     7EA2 7D45
0115 7EA4 1601   JNE  >7EA8
0116 7EA6 0587   INC  R7
0117 7EA8 05C3   INCT R3
0118 7EAA 0283   CI   R3,16
     7EAC 0010
0119 7EAE 16F3   JNE  >7E96
0120 7EB0 2620   CZC  @>7D5E,R8
     7EB2 7D5E
0121 7EB4 130F   JEQ  >7ED4
0122 7EB6 0287   CI   R7,1
     7EB8 0001
0123 7EBA 1501   JGT  >7EBE
0124 7EBC 1008   JMP  >7ECE
0125 7EBE 0287   CI   R7,3
     7EC0 0003
0126 7EC2 1505   JGT  >7ECE
0127 7EC4 05A0   INC  @>7D0E
     7EC6 7D0E
0128 7EC8 DDA0   MOVB @>7D44,*R6+
     7ECA 7D44
0129 7ECC 100D   JMP  >7EE8
0130 7ECE DDA0   MOVB @>7D46,*R6+
     7ED0 7D46
0131 7ED2 100A   JMP  >7EE8
0132 7ED4 0287   CI   R7,3
     7ED6 0003
0133 7ED8 1605   JNE  >7EE4
0134 7EDA 05A0   INC  @>7D0E
     7EDC 7D0E
0135 7EDE DDA0   MOVB @>7D45,*R6+
     7EE0 7D45
0136 7EE2 1002   JMP  >7EE8
0137 7EE4 DDA0   MOVB @>7D47,*R6+
     7EE6 7D47
0138 7EE8 05C0   INCT R0
0139 7EEA 0280   CI   R0,>7CEA
     7EEC 7CEA
0140 7EEE 16CB   JNE  >7E86
0141            * balayage clavier
0142 7EF0 0300   LIMI 2
     7EF2 0002
0143 7EF4 0300   LIMI 0
     7EF6 0000
0144 7EF8 0420   BLWP @>6020
     7EFA 6020
0145 7EFC 9820   CB   @>8375,@>7D4A
     7EFE 8375
     7F00 7D4A
0146 7F02 13F6   JEQ  >7EF0
0147 7F04 9820   CB   @>8375,@>7D4C
     7F06 8375
     7F08 7D4C
0148 7F0A 1602   JNE  >7F10
0149 7F0C 0460   B    @>7DDA
     7F0E 7DDA
0150 7F10 9820   CB   @>8375,@>7D4B
     7F12 8375
     7F14 7D4B
0151 7F16 16EC   JNE  >7EF0
0152            * transferts tampons
0153 7F18 0200   LI   R0,>2040
     7F1A 2040
0154 7F1C 0201   LI   R1,>774A
     7F1E 774A
0155 7F20 0202   LI   R2,>7A0A
     7F22 7A0A
0156 7F24 D800   MOVB R0,@>8C02
     7F26 8C02
0157 7F28 06C0   SWPB R0
0158 7F2A D800   MOVB R0,@>8C02
     7F2C 8C02
0159 7F2E 1000   JMP  $+2
0160 7F30 D831   MOVB *R1+,@>8C00
     7F32 8C00
0161 7F34 8081   C    R1,R2
0162 7F36 16FC   JNE  >7F30
0163 7F38 05A0   INC  @>7D0C
     7F3A 7D0C
0164 7F3C 8820   C    @>7D0C,@>7D60
     7F3E 7D0C
     7F40 7D60
0165 7F42 1602   JNE  >7F48
0166 7F44 0460   B    @>7DDA
     7F46 7DDA
0167 7F48 0200   LI   R0,>7A2A
     7F4A 7A2A
0168 7F4C 0201   LI   R1,>774A
     7F4E 774A
0169 7F50 CC31   MOV  *R1+,*R0+
0170 7F52 0280   CI   R0,>7CEA
     7F54 7CEA
0171 7F56 16FC   JNE  >7F50
0172            * affiche nombre
0173            * de generations
0174 7F58 C160   MOV  @>7D0C,R5
     7F5A 7D0C
0175 7F5C 06A0   BL   @>7F78
     7F5E 7F78
0176 7F60 02EC   DATA >02EC
0177 7F62 0460   B    @>7E70
     7F64 7E70
0178            * sous-programme
0179            * nombres aleatoires
0180 7F66 0204   LI   R4,28645
     7F68 6FE5
0181 7F6A 3920   MPY  @>83C0,R4
     7F6C 83C0
0182 7F6E 0225   AI   R5,31417
     7F70 7AB9
0183 7F72 C805   MOV  R5,@>83C0
     7F74 83C0
0184 7F76 045B   B    *R11
0185            * sous-programme
0186            * affichage nombres
0187 7F78 C03B   MOV  *R11+,R0
0188 7F7A C80B   MOV  R11,@>7D0A
     7F7C 7D0A
0189 7F7E 04C4   CLR  R4
0190 7F80 3D20   DIV  @>7D62,R4
     7F82 7D62
0191 7F84 0224   AI   R4,>30
     7F86 0030
0192 7F88 D060   MOVB @>7D19,R1
     7F8A 7D19
0193 7F8C 0420   BLWP @>6024
     7F8E 6024
0194 7F90 0580   INC  R0
0195 7F92 04C4   CLR  R4
0196 7F94 3D20   DIV  @>7D64,R4
     7F96 7D64
0197 7F98 0224   AI   R4,>30
     7F9A 0030
0198 7F9C D060   MOVB @>7D19,R1
     7F9E 7D19
0199 7FA0 0420   BLWP @>6024
```

```
     7FA2 6024
     7FA4 0580   INC  R0
     7FA6 0225   AI   R5,>30
     7FA8 0030
     7FAA D060   MOVB @>7D1B,R1
     7FAC 7D1B
     7FAE 0420   BLWP @>6024
     7FB0 6024
     7FB2 C2E0   MOV  @>7D0A,R11
     7FB4 7D0A
     7FB6 045B   B    *R11
                 *
     701C        AORG >701C
     701C 7FB8   DATA >7FB8
     701E 7FE0   DATA >7FE0
     7FE0        AORG >7FE0
     7FE0   4A   TEXT 'JEUVIE'
     7FE6 7D82   DATA >7D82
                 *
                      END
```

Programmathèque

Marianne Sutz

Cette rubrique a pour but de porter à votre connaissance les nouveaux programmes proposés pour notre ordinateur.

Burger Time

Ce jeu a été conçu par *Data East USA, Inc.*, et est commercialisé par *Texas Instruments.*

Le scénario est assez loufoque : un cuisinier, spécialisé dans les hamburgers, est poursuivi par des saussices, des cornichons et des oeufs et ne dispose, pour se défendre, que d'un poivrier qui lui permet d'endormir ses agresseurs.
En marchant sur les éléments constituant ses hamburgers (tranches de pain, fromage, etc...), il peut aussi les faire tomber et ainsi écraser les monstres comestibles.

Le jeu est intéressant et le graphisme très soigné (utilisation du mode "bit map"). Il faut tout de même lui faire un reproche : il est affublé d'une musique omniprésente et répétitive qui devient rapidement insupportable et nous oblige à couper le son du téléviseur.

Star Trek

Créé par *Sega Enterprises Incorporated*, et commercialisé par *Texas Instruments.*

C'est certainement le jeu le plus intéressant que nous ayons essayé. Il est inspiré du célèbre feuilleton américain produit par *Paramount Pictures Corporation* et peut utiliser le synthétiseur de paroles.

Vous êtes le capitaine du vaisseau spacial "Enterprise" et avez pour mission de protéger la fédération galactique. Vos adversaires sont les Klingons et les nomades de l'espace. Pour combattre ces envahisseurs, vous disposez de "torpilles photon" et de "phaseurs".

L'écran se présente de la manière suivante :

- dans le coin supérieur gauche se trouvent des indicateurs : réserve d'énergie pour le bouclier de protection, réserve d'énergie pour les déplacements rapides et nombres de "torpilles photon" disponibles;
- la partie restante en haut et à droite de l'écran représente une vue aérienne de la zone de combat, ce qui vous permet de situer vos adversaires;
- la partie inférieure (la plus vaste) simule, en trois dimensions, ce que vous pouvez voir depuis votre poste de pilotage.

Un très bon programme qui passionnera certainement ceux qui ont apprécié *Parsec.*

Hopper

Ce module est produit et commercialisé par *Texas Instruments.*

Trois chasseurs essaient de capturer un kangourou, pour le vendre à un cirque. La scène se passe dans la cale d'un cargo dans laquelle sont réparties plusieurs cages. Pour se défendre, le kangourou doit pousser les cages pour enfermer les chasseurs.

Il faut être franc; ce jeu ne nous a pas enthousiasmé. Le graphisme est sommaire, et il nous semble qu'un amateur peut faire aussi bien en peu de temps.

Return to pirate's isle

Créé par *Scott Adams*, et commercialisé par *Texas Instruments.*

Il faut mettre en garde les acheteurs potentiels de ce programme; il s'agit d'un jeu de rôle de très haut niveau et le graphisme est remarquable, malheureusement, toutes les indications fournies au programme doivent être en Anglais. Si vous maîtrisez parfaitement cette langue, aucun problème. Sinon, cet excellent programme est inutilisable.

Treasure Island

Un jeu imaginé par *Data East USA*, et commercialisé par *Texas Instruments.*

Un aventurier cherche des trésors dans une île peuplée de gorilles monstrueux, qui essaient de vous attraper et vous lancent des pierres pour vous faire tomber à l'eau.
Pour compliquer les choses, l'île est en train de s'enfoncer dans la mer.

Ce jeu aurait pu être très bon, d'autant plus que les images générées sont de bonne qualité, mais la part laissée au hasard nous semble trop importante.

Les jeux Atarisoft

Atarisoft, la division "logiciels" de la célèbre firme *Atari*, propose quatre programmes fonctionnant sur le TI-99/4A : *PAC-MAN, JUNGLE HUNT, MOON PATROL et DEFENDER.*

Nous n'avons pas encore pu tester ces jeux, mais nous en reparlerons dès que nous pourrons nous les procurer.

Lunar Lander 2

Enfin un programme français. Ce programme, écrit en Basic Etendu et disponible sur cassette, est commercialisé par le *Club de Micro-informatique de Chatou* - 72 rue des landes - 78400 Chatou, au prix de 95 FF.

Nous étions en train de boucler ce numéro lorsque ce programme est parvenu jusqu'à nos bureaux. Nous n'avons donc pas eu le temps de le tester. Nous en reparlerons dans le prochain numéro. En attendant, nous ne pouvons que féliciter les membres du club pour cette excellente initiative. ■

Les opérateurs logiques

Julien Thomas

Cet article constitue la seconde partie de la série consacrée aux particularités du Basic Etendu. Toutefois, les opérateurs logiques étant largement employés en assembleur, les explications fournies ci-dessous pourront être utiles à ceux qui débutent avec ce langage.
Il sera très souvent fait appel au système de numération en base 2 (binaire); nous vous conseillons donc de consulter l'article intitulé "Conversions", publié dans ce numéro, avant de poursuivre la lecture de cet article.

Nous nous sommes aperçus que les opérateurs logiques, lorsqu'ils sont présents dans un programme écrit en Basic Etendu, sont très souvent utilisés en tant que "répétiteurs de tests", alors qu'il peuvent aussi être employés de manière totalement différente. Ceci est certainement dû au fait que le manuel du Basic Etendu est assez discret sur ce point.

Les opérateurs logiques, appelés aussi expressions booléennes (du nom du mathématicien Boole, qui créa une algèbre portant sur des variables ne pouvant prendre que deux valeurs : 1 ou 0, vrai ou faux) opèrent directement sur la représentation binaire du ou des nombres à traiter. Les opérateurs disponibles en Basic Etendu sont : AND (et), OR (ou), XOR (ou exclusif) et NOT (non).

Les paragraphes suivants décrivent le fonctionnement de ces fonctions. Plutôt que de faire de nombreux schémas figés, nous avons préféré écrire un petit programme simulant le travail effectué par le microprocesseur. Il convient donc de charger ce programme dès à présent.

Vous vous rendrez aisément compte que le programme n'accepte que des entiers compris entre −32768 et 65535. Ceci correspond aux valeurs maximales qu'il est possible de coder en binaire sur 16 bits, puisque le microprocesseur utilise ce nombre de bits pour réaliser les opérations logiques.

AND (ET)

Le AND (ou *intersection logique*) place à 1 le bit résultant si les bits correspondant des deux nombres sont à 1, à 0 dans le cas contraire :
1 AND 1 = 1
1 AND 0 = 0
0 AND 1 = 0
0 AND 0 = 0

Autrement dit, le bit résultant de l'opération est placé à 1 uniquement si le bit du premier nombre ET le bit du deuxième nombre sont à 1. Ainsi, si l'on effectue un AND entre 65535 et 256 :

65535 = 1111111111111111
256 = 0000000100000000
65535 AND 256 =000000010000 0000 = 256.

Ce type d'opération peut être utilisé pour déterminer l'état de un ou plusieurs bits donnés. Dans l'article *"Les lutins du Basic Etendu"*, publié dans le numéro 4 de "99 Magazine", cette méthode était employée pour détecter la présence de plus de quatre lutins sur la même ligne.

OR (ou)

Le OR *(réunion logique)* place un 1 dans le résultat, si un des deux bits correspondant à un rang donné dans les nombres traités est à 1. Donc :

1 OR 1 = 1
1 OR 0 = 1
0 OR 1 = 1
0 OR 0 = 0

Concrètement, 1234 OR 5678 se traduit par :

1234 = 0000010011010010
5678 = 0001011000101110
1234 OR 5678 =0001011011111 110 = 5886.

Si cela ne vous paraît pas très clair, essayez quelques opérations logiques avec le programme proposé dans cet article; il simule le traitement bit à bit et le système utilisé deviendra transparent.

XOR (ou exclusif)

Cet opérateur réalise une *disjonction logique* entre deux nombres, et fonctionne à peu près de la même façon que OR mais seulement si l'un des deux bits est à 1 (l'un ou l'autre, mais pas les deux) :

1 XOR 1 = 0
1 XOR 0 = 1
0 XOR 1 = 1
0 XOR 0 = 0

NOT (non)

Le NOT est le *complément à 1* du nombre traité (cet opérateur ne requiert qu'un seul argument). Ceci revient à inverser la valeur de tous les bits constituant le nombre :

NOT 1 = 0
NOT 0 = 1

NOT 34 (0000000000100010) est égal à 1111111111011101.

Vous pouvez obtenir plus de renseignements sur le *complément à un* en consultant l'article "Conversions".

Les opérateurs logiques sont des outils qu'il ne faut pas négliger, et qui peuvent parfois faire gagner pas mal de place en mémoire.

Par exemple, certains programmes utilisent de nombreux drapeaux. Dans la plupart des cas, une variable est affectée à chaque drapeau, ce qui fait que, si l'on en emploie seize, 16*8 soit 128 octets (une variable numérique utilise 8 octets) sont "consommés" uniquement pour ces indicateurs.

Les opérateurs logiques permettent le traitement d'un nombre bit par bit. Il est donc possible, avec des ORs et des ANDs, de se servir d'une seule variable pour 16 drapeaux, et d'économiser ainsi 120 octets : si Y est le nom de la variable réservée aux indicateurs, Y = Y OR 1 placera le bit de poids faible de Y à 1, sans mofifier l'état des autres bits. Inversement, Y = Y AND 65534 forcera le bit de poids faible à 0, toujours sans modifier les autres. Y = Y XOR 1 place le bit de poids faible de Y à 1 uniquement si ce bit est à 0. Pour tester l'état du bit de poids faible de Y, il suffit de faire D = Y AND 1. Si D est égal à 0, le bit de poids faible est à 0, sinon, il est à 1. Dans un programme, on peut écrire :

100 IF Y AND 1 THEN DRAPEAU1=1 ELSE DRAPEAU1=0

Ceci est un exemple d'application mais nous aurions pu en trouver beaucoup d'autres. Le but de cet article était surtout de vous faire comprendre le fonctionnement de ces divers opérateurs qui, nous l'espérons, trouveront leur place dans vos futures réalisations. ■

```
100 ! ***********************
110 ! *   Les  Operateurs   *
120 ! *      Logiques       *
130 ! ***********************
140 ! *    Basic  Etendu    *
150 ! ***********************
160 !
170 OPTION BASE 1 :: DIM O$(4):: CALL MAG
NIFY(4):: CALL CLEAR :: CALL SCREEN(5)
180 IMAGE ###### =
```

```
190 DATA FC84848484FC,0,0,0,FF80FF,FF00FF
,FF01FF,OR,AND,XOR,NOT
200 GOTO 220 :: CALL CHAR :: CALL CHARPAT
 :: CALL COLOR :: CALL DELSPRITE :: CAL
L HCHAR :: CALL KEY :: CALL SPRITE
210 B :: D :: DR :: K :: N :: N1 :: N2 ::
 N3 :: NO :: NX :: S :: C$ :: C1$ :: C2
$ :: N$ :: NB$ :: NB1$ :: NB2$ :: OE$ ::
!@P-
220 FOR B=0 TO 12 :: CALL COLOR(B,16,1)::
 NEXT B :: CALL COLOR(14,4,1,13,10,1)
230 FOR B=136 TO 142 :: READ C$ :: CALL C
HAR(B,C$):: NEXT B :: CALL CHARPAT(48,C
1$,49,C2$):: CALL CHAR(128,C1$,129,C2$)
240 FOR B=1 TO 4 :: READ O$(B):: NEXT B
250 DISPLAY AT(1,3)ERASE ALL:"OPERATEUR ?
 :"
260 ACCEPT AT(1,17)VALIDATE("ADNORTX")BEE
P SIZE(-3):OE$ :: DR=1 :: FOR B=1 TO 4
:: IF OE$=O$(B)THEN DR=0 :: NO=B
270 NEXT B :: IF DR THEN 260 ELSE IF NO=4
 THEN 500 ELSE CALL CHAR(132,"FC8484848
48484848484848484FC0000000000000000000000000"
)
280 DISPLAY AT(3,3):"NOMBRE 1 ? :"
290 ACCEPT AT(3,16)VALIDATE(DIGIT,"-")BEE
P SIZE(-6):N$ :: DR=0 :: FOR B=2 TO LEN
(N$):: IF SEG$(N$,B,1)="-" THEN DR=1
300 NEXT B :: IF DR OR N$="" THEN 290 ELS
E N1=VAL(N$):: IF N1<-32768 OR N1>32767
 THEN 290 ELSE DISPLAY AT(5,3):"NOMBRE 2
? :"
310 ACCEPT AT(5,16)VALIDATE(DIGIT,"-")BEE
P SIZE(-6):N$ :: DR=0 :: FOR B=2 TO LEN
(N$):: IF SEG$(N$,B,1)="-" THEN DR=1
320 NEXT B :: IF DR OR N$="" THEN 310 ELS
E N2=VAL(N$):: IF N2<-32768 OR N2>32767
 THEN 310
330 DISPLAY AT(8,3):USING 180:N1 :: NX=N1
 :: IF NX<0 THEN NX=NX+65536
340 GOSUB 610 :: DISPLAY AT(8,12):NB$ ::
NB1$=NB$
350 DISPLAY AT(10,3):USING 180:N2 :: NX=N
2 :: IF NX<0 THEN NX=NX+65536
360 GOSUB 610 :: DISPLAY AT(10,12):NB$ ::
 NB2$=NB$
370 CALL HCHAR(12,14,140):: CALL HCHAR(12
,15,141,14):: CALL HCHAR(12,29,142)
380 FOR B=1 TO 16 :: CALL SPRITE(#1,132,4
,55,95+B*8,#2,136,10,95,95+B*8)
390 C1$=SEG$(NB1$,B,1):: C2$=SEG$(NB2$,B,
1):: ON NO GOSUB 630,650,670 :: CALL HC
HAR(13,13+B,ASC(C$))
400 DISPLAY AT(15,12)BEEP:C1$&" "&O$(NO)&
" "&C2$&" = "&CHR$(ASC(C$)+80):: FOR D=
0 TO 150 :: IF INT(D/10)<>D/10 THEN 420
410 IF DR THEN CALL COLOR(#2,14):: CALL C
OLOR(13,14,1):: DR=0 ELSE CALL COLOR(#2
,10):: CALL COLOR(13,10,1):: DR=1
420 NEXT D :: NEXT B
430 CALL HCHAR(15,14,32,16):: ON NO GOTO
440,450,460
440 N3=N1 OR N2 :: GOTO 470
450 N3=N1 AND N2 :: GOTO 470
460 N3=N1 XOR N2
470 CALL DELSPRITE(ALL):: DISPLAY AT(13,3
)BEEP SIZE(8):USING 180:N3 :: DISPLAY A
T(17,3):"donc, ";N1;O$(NO);N2 :: DISPLAY
AT(18,3):"est egal a ";N3
480 DISPLAY AT(20,3):"Autre operation ? (
O/N)"
490 CALL KEY(0,K,S):: IF K=79 THEN 250 EL
SE IF K=78 THEN CALL CLEAR :: END ELSE
490
500 CALL CHAR(132,"0000000000000000FC8484
8484FC000000000000000000000"):: DISPLAY
 AT(3,3):"NOMBRE ? :"
510 ACCEPT AT(3,14)VALIDATE(DIGIT,"-")BEE
P SIZE(-6):N$ :: DR=0 :: FOR B=2 TO LEN
(N$):: IF SEG$(N$,B,1)="-" THEN DR=1
520 NEXT B :: IF DR OR N$="" THEN 510 ELS
E N=VAL(N$):: IF N<-32768 OR N>32767 TH
EN 510
530 DISPLAY AT(10,3):USING 180:N :: NX=N
:: IF NX<0 THEN NX=NX+65536
540 GOSUB 610 :: DISPLAY AT(10,12):NB$ ::
 CALL HCHAR(12,14,140):: CALL HCHAR(12,
15,141,14):: CALL HCHAR(12,29,142)
550 FOR B=16 TO 1 STEP -1 :: CALL SPRITE(
#1,132,4,55,95+B*8,#2,136,10,95,95+B*8)
560 C$=SEG$(NB$,B,1):: IF C$="0" THEN NX=
49 ELSE NX=48
570 CALL HCHAR(13,13+B,NX):: DISPLAY AT(1
5,12)BEEP:C$&" devient "&CHR$(NX+80)::
FOR D=0 TO 150 :: IF INT(D/10)<>D/10 THEN
 590
580 IF DR THEN CALL COLOR(#2,14):: CALL C
OLOR(13,14,1):: DR=0 ELSE CALL COLOR(#2
,10):: CALL COLOR(13,10,1):: DR=1
590 NEXT D :: NEXT B :: DISPLAY AT(13,3)B
EEP SIZE(8):USING 180:NOT N :: CALL HCH
AR(15,14,32,11):: CALL DELSPRITE(ALL)
600 DISPLAY AT(17,3):"donc, NOT ";N :: DI
SPLAY AT(18,3):"est egal a ";NOT N :: G
OTO 480
610 NB$="" :: FOR B=15 TO 0 STEP -1 :: IF
 NX<2^B THEN NB$=NB$&"0" ELSE NX=NX-2^B
 :: NB$=NB$&"1"
620 NEXT B :: RETURN
630 IF C1$="1" OR C2$="1" THEN C$="1" ELS
E C$="0"
640 RETURN
650 IF C1$="1" AND C2$="1" THEN C$="1" EL
SE C$="0"
660 RETURN
670 IF C1$="1" XOR C2$="1" THEN C$="1" EL
SE C$="0"
680 RETURN
```

Apprendre le Basic en programmant

Roger Didi

L'alibi du jour sera la construction d'une grille de mots croisés.

L'utilisateur nous demande le dessin d'une grille carrée dont il pourra choisir les dimensions (moins de 15 cases par ligne). Ces cases seront repérées par deux lettres : la case C,D se trouve donc au croisement de la ligne horizontale C et de la ligne verticale D.

Il voudrait pouvoir placer à son gré des cases noires, et essayer ensuite de croiser des mots; la machine devra lui signaler toute erreur de croisement et de position.

Nous conseillons donc au lecteur de se munir d'une grille 32*24 qui représente l'écran car nous aurons plusieurs petits problèmes de positionnement de caractères à résoudre.

Programme 1 : placer les cases noires

Problèmes à résoudre :

1) dessiner la grille;
2) écrire un texte au-dessous de la grille;
3) convertir les coordonnées d'une case de la grille en position sur l'écran (pour utiliser CALL HCHAR).

Dessin de la grille

Vous créez, à l'aide de CALL CHAR, le caractère numéro 100; c'est un petit carré auquel vous donnez des couleurs avec CALL COLOR (il appartient au groupe 9). Créez aussi le caractère 101 (même couleur) qui correspond à la case noire. Choisissez une couleur pour l'écran en utilisant CALL SCREEN, et une couleur pour les lettres (groupes 5 à 7); enfin, nettoyez l'écran (CALL CLEAR).

```
10 REM *********************
20 REM *DESSIN DE LA GRILLE*
30 REM *********************
100 CALL CLEAR
110 CALL COLOR(9,9,16)
120 CALL SCREEN(3)
130 CALL CHAR(100,"FF81818181818
1FF")
140 FOR I=6 TO 16
150 CALL HCHAR(I,11,100,10)
160 NEXT I
170 GOTO 170
```

Il faut encore prévoir deux choses :

- un tableau (que nous nommons JEU(X,Y)), pour mettre en réserve, et suivre, les codes ASCII des caractères placés dans la grille. L'instruction DIM(16,16) permet de réserver une place dans la mémoire pour ce tableau, qui permettra de suivre ce qui ce passe sur la grille. De plus, lorsque la grille est pleine, l'utilisateur peut souhaiter la conserver sur cassette ou sur disquette; la conservation de ce tableau suffira pour cela;
- un tableau (que nous nommons ATT(X)) pour écrire le mot (en attente) que l'utilisateur écrit à côté de la grille avant que la machine ne le reporte dans celle-ci. Cela nous permettra de nous assurer de l'absence d'erreur dans la grille.

Maintenant, demandez la dimension de la grille à l'aide de INPUT et placez la longueur du côté dans une variable (que nous nommons C).

Il faut tester C et refuser un nombre inférieur à 1 ou supérieur à 15.

La grille doit être placée au milieu de l'écran. Il faut donc calculer, en fonction de C, la position de chaque carré. D'autre part, il faut écrire le nom de chaque ligne (à gauche) et le nom de chaque colonne (au-dessus). Enfin, il faut remplir le tableau JEU(X,Y) avec le nombre 100. Nous saurons qu'une case est blanche si nous trouvons 100 dans la position correspondante du tableau JEU.

Ecrire un texte

Ce texte doit être écrit au-dessous de la grille, donc aux lignes 21 à 24 de l'écran; il ne doit pas provoquer un "scroll" (ce qui ferait monter la grille). Nous avons construit une routine de même nature dans le numéro précédent de "99 Magazine". Il doit comporter trois lignes :

SITUATION DES NOIRES
ENTER SI PLUS DE NOIRE
LIGNE,COLONNE --->

En effet, il faut prévoir une sortie de cette partie du programme lorsque l'utilisateur le souhaitera; le code de la touche "ENTER" est 13.

Rappelons que CALL KEY(0,K,S) permet une scrutation du clavier. La variable S prend la valeur 0 (zéro) quand aucune touche n'est pressée. Il suffira donc de tester S et K pour savoir ce qui se passe.

Convertir les coordonnées

L'utilisateur désigne sa case par deux lettres. Nous lui éviterons la virgule qui se placera dès qu'il aura écrit la première coordonnée. Il faudra l'empêcher d'utiliser une lettre qui ne correspond pas à la grille.

Nous aurons alors un gros travail à faire pour placer la case noire : il s'agit de calculer la position de cette case sur l'écran et dans le tableau JEU.

Dans jeu

NH=KY−64
NV=KX−64

En effet, le code ASCII de A est 65 et il correspond à la position 1 sur la grille.

Sur l'écran

Il faut ajouter 12−CC ou 15−CC pour retrouver la position sur l'écran (CC est la moitié du côté du carré).

LIGNE=KY−64+12−CC
COL=KX−64+15−CC

Il n'y a plus qu'à placer 101 dans JEU(NH,NV) et dessiner une case noire avec CALL HCHAR(LIGNE, COL,101).

Cette routine de calcul des coordonnées mérite toute votre attention; elle sera utile dans tout programme utilisant un repérage de l'écran par l'utilisateur.

N.D.L.R. : Le programme listé ci-dessous représente la première partie du programme définitif. Il n'est donc pas autonome et ne peut être utilisé que pour le positionnement des cases noires. Le fait de vouloir aller plus loin provoquerait l'affichage d'un message d'erreur.

```
100 REM *****************
110 REM * PROGRAMME No1   *
120 REM *  CASES NOIRES   *
130 REM *****************
140 REM *    BASIC TI     *
150 REM *****************
160 REM *   COPYRIGHT     *
170 REM *   ROGER DIDI    *
180 REM * ET 99 MAGAZINE *
190 REM *****************
200 REM
210 DIM JEU(16,16)
220 DIM ATT(15)
230 CALL CLEAR
240 CALL COLOR(9,9,16)
250 FOR I=2 TO 8
260 CALL COLOR(I,2,11)
270 NEXT I
280 CALL SCREEN(3)
```

```
290 CALL CHAR(100,"FF81818181818
1FF")
300 CALL CHAR(101,"FFFFFFFFFFFF
FFF")
310 REM *******************
320 REM DESSIN DE LA GRILLE
330 REM ******************
340 INPUT "COTE DU CARRE (C<=15)
 ? ":C
350 IF (C<1)+(C>15)=-1 THEN 340
360 CC=INT(C/2)
370 CALL CLEAR
380 FOR I=1 TO C
390 CALL HCHAR(12-CC+I,16-CC-2,6
4+I)
400 CALL HCHAR(10-CC,15-CC+I,64+
I)
410 CALL HCHAR(12-CC+I,16-CC,100
,C)
420 FOR J=1 TO C
430 JEU(I,J)=100
440 NEXT J
450 NEXT I
460 REM ******************
470 REM PLACER LES NOIRES
480 REM ******************
490 MOT$="SITUATION DES NOIRES"
500 Y=21
510 X=3
520 GOSUB 650
530 MOT$="('ENTER' SI PLUS DE NO
IRE)"
540 Y=22
550 GOSUB 650
560 GOSUB 740
570 IF KY=13 THEN 32767
575 REM ADRESSE A CORRIGER PAR L
A SUITE
580 K=101
590 JEU(NH,NV)=101
600 CALL HCHAR(LIGNE,COL,K)
610 GOTO 560
620 REM *******************
630 REM ECRITURE D'UN MOT SUR L'
ECRAN
640 REM *******************
650 L=LEN(MOT$)
660 FOR U=1 TO L
670 K=ASC(SEG$(MOT$,U,1))
680 CALL HCHAR(Y,X+U,K)
690 NEXT U
700 RETURN
710 REM *******************
720 REM LECTURE DES COORDONNEES
730 REM *******************
740 MOT$="(LIGNE,COLONNE)-->
 "
750 X=3
760 Y=23
770 GOSUB 650
780 CALL KEY(0,KY,S)
790 IF S=0 THEN 780
800 IF KY=13 THEN 910
810 IF (KY<65)+(KY>64+C)=-1 THEN
 780
820 CALL HCHAR(23,22,KY)
830 CALL HCHAR(23,24,ASC(","))
840 CALL KEY(0,KX,S)
850 IF (KX<65)+(KX>64+C)=-1 THEN
 840
860 CALL HCHAR(23,26,KX)
870 LIGNE=KY-64+12-CC
880 COL=KX-64+15-CC
890 NH=KY-64
900 NV=KX-64
910 RETURN
```

Comparez donc ce programme avec le vôtre, améliorez-le et essayez de prévoir une suite avant de continuer avec nous.

Programme 2 : mots croisés

Nous avons choisi d'écrire moins de programmes pour éviter de remplir inutilement les pages de "99 Magazine", mais nous conseillons au lecteur de réaliser lui-même ce programme morceau par morceau.

Première étape

L'utilisateur a pressé la touche "ENTER" et se prépare donc à écrire des mots. Ceux-ci peuvent être écrits verticalement ou horizontalement : un texte doit donc être écrit au-dessous de la grille pour demander à l'utilisateur son choix; nous choisissons

ECRITURE DES MOTS
HORIZONTALEMENT---> H
VERTICALEMENT ----> V

Nous disposons d'une routine d'écriture et ceci ne doit pas poser de difficulté. Cependant, il faudrait prévoir une routine qui efface le texte sans toucher à la grille. Construisez donc cette partie du programme et essayez-la.

Deuxième étape

Un CALL KEY(0,KK,S) doit permettre de placer le choix de l'utilisateur dans la variable KK. Il peut être ici intéressant d'utiliser les drapeaux dont nous avons parlé dans un numéro précédent de "99 Magazine". Il s'agit de variables qui conserveront le choix de l'utilisateur et que nous pourrons tester continuellement.

ZH prend la valeur 1 si "H" est pressé, la valeur 0 dans le cas contraire;
ZV prend la valeur 1 si "V" est pressé, la valeur 0 dans le cas contraire.

Ceci se fait aisément :
ZH=−(KK=ASC("H")) et ZV=−(KK=ASC("V"))
En effet, A=B vaut (−1) si vrai et 0 si faux.

Troisième étape

Le texte :
ENTER SI MOT TERMINE
LIGNE,COLONNE--->
doit apparaître au bas de l'écran et la demande de coordonnée utilise la routine que nous avons déjà construite; ceci ne doit donc pas vous poser de problème. A la fin de cette étape, nous devons disposer des coordonnées sur l'écran de la première lettre du mot et des coordonnées de la même lettre dans le tableau JEU.

Quatrième étape

Des erreurs peuvent maintenant se produire : il faut donc construire une routine musicale ou autre, signalant un incident à l'utilisateur. A chaque erreur, cette routine enverra un message et l'utilisateur devra recommencer.

Quelles sont les erreurs possibles ?

1) la case choisie n'est pas acceptable : elle n'est pas précédée d'une noire ou elle n'est pas en début de ligne ou de colonne.

Voyons ici l'intérêt des drapeaux :
NH−ZV,NV−ZH désigne la case précédente dans les deux cas. En effet, si l'écriture est horizontale, ZH=1 et ZV=0, sinon ZV=1 et ZH=0. Il suffit donc de tester JEU(NH−ZV,NV−ZH) !!
Intéressez-vous donc à la ligne 790 du programme pour bien comprendre le fonctionnement des tests.

2) si le choix de la position du mot est bon, l'utilisateur peut commencer à écrire; mais il peut commettre des erreurs :

- la case est occupée par une noire;
- la case est occupée par une lettre différente.

Autrement dit, la case n'est pas blanche ou le code de la lettre proposée est différent de celui qui est en mémoire. Il faut donc utiliser ici un compteur de lettres (COMPT) et la réserve ATT() que nous avons préparée.

Ici encore, les drapeaux vont servir.
JEU(NH+COMPT*ZV,NV+COMPT*ZH) désigne la case suivante dans la direction choisie (essayez de comprendre pourquoi). Il suffit donc de tester cette case.

Avant ces tests, il faut permettre l'écriture des essais dans une zone de l'écran : nous avons choisi la zone à droite de la grille et verticalement pour les mots verticaux, et au-dessus de la grille pour les mots horizontaux (ceci doit expliquer aussi la routine d'effacement).

3) enfin, il faut surveiller le dépassement des bords de la grille.
Lorsque l'utilisateur a terminé son mot et que celui-ci est correct, il suffit de le recopier dans la grille et dans le tableau JEU en utilisant toujours les drapeaux; nous laissons au lecteur le soin de le faire avant de lire notre programme.

```
10 REM *****************
15 REM * PROGRAMME FINAL*
20 REM *  MOTS CROISES  *
25 REM *****************
30 REM *    BASIC TI    *
35 REM *****************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM *****************
60 REM
100 DIM JEU(16,16)
110 DIM ATT(15)
120 CALL CLEAR
130 CALL COLOR(9,9,16)
140 FOR I=2 TO 8
150 CALL COLOR(I,2,11)
160 NEXT I
170 CALL SCREEN(3)
180 CALL CHAR(100,"FF818181818181FF")
190 CALL CHAR(101,"FFFFFFFFFFFFFFF")
200 REM *******************
210 REM DESSIN DE LA GRILLE
220 REM *******************
230 INPUT "COTE DU CARRE (C<=15) ? ":C
240 IF (C<1)+(C>15)=-1 THEN 230
250 CC=INT(C/2)
260 CALL CLEAR
270 FOR I=1 TO C
280 CALL HCHAR(12-CC+I,16-CC-2,64+I)
290 CALL HCHAR(10-CC,15-CC+I,64+I)
300 CALL HCHAR(12-CC+I,16-CC,100,C)
310 FOR J=1 TO C
320 JEU(I,J)=100
330 NEXT J
340 NEXT I
350 REM ******************
360 REM PLACER LES NOIRES
370 REM ******************
380 MOT$="SITUATION DES NOIRES"
390 Y=21
400 X=3
410 GOSUB 1150
420 MOT$="('ENTER' SI PLUS DE NOIRE)"
430 Y=22
440 GOSUB 1150
450 GOSUB 1240
460 IF KY=13 THEN 540
470 K=101
480 JEU(NH,NV)=101
490 CALL HCHAR(LIGNE,COL,K)
500 GOTO 450
510 REM ******************
520 REM CHOIX DE DIRECTION
530 REM ******************
540 GOSUB 1530
550 MOT$="ECRITURE DES MOTS"
560 Y=21
570 X=3
580 GOSUB 1150
590 MOT$="HORIZONTALEMENT ---> H"
600 Y=22
610 X=3
620 GOSUB 1150
630 MOT$="VERTICALEMENT------> V"
640 Y=23
650 COMPT=0
660 GOSUB 1150
670 CALL KEY(0,KK,S)
680 IF (KK<>ASC("H"))+(KK<>ASC("V"))=-2 THEN 670
690 ZH=-(KK=ASC("H"))
700 ZV=-(KK=ASC("V"))
710 REM ******************
720 REM PLACER LES MOTS
730 REM ******************
740 GOSUB 1530
750 MOT$="('ENTER' SI MOT TERMINE)"
760 Y=22
770 GOSUB 1150
780 GOSUB 1240
790 IF (JEU(NH-ZV,NV-ZH)=101)+(NH=1)*ZV+(NV=1)*ZH THEN 820
800 GOSUB 1450
810 GOTO 780
820 MOT$="MOT V."
830 IF ZV=1 THEN 850
840 MOT$="MOT H."
850 GOSUB 1570
860 X=24*ZV+3*ZH
870 Y=6*ZV+2*ZH
880 GOSUB 1150
890 CALL KEY(0,K,S)
900 IF S=0 THEN 890
910 IF ((K<65)+(K>90))*(K<>13)THEN 890
920 CASE=JEU(NH+COMPT*ZV,NV+COMPT*ZH)
930 IF K=13 THEN 1040
940 IF (CASE=100)+(CASE=K)THEN 980
950 GOSUB 1450
960 GOSUB 1570
970 GOTO 540
980 ATT(COMPT)=K
990 COMPT=COMPT+1
1000 CALL HCHAR(Y*ZV+2,27*ZV+(X+9)*ZH,K)
1010 Y=Y+1
1020 X=X+1
1030 GOTO 890
1040 IF (CASE<>101)+(NH*ZV+NV*ZH+COMPT<>C+1)=-2 THEN 950
1050 FOR I=1 TO COMPT
1060 ZZV=(I-1)*ZV
1070 ZZH=(I-1)*ZH
1080 JEU(NH+ZZV,NV+ZZH)=ATT(I-1)
1090 CALL HCHAR(LIGNE+ZZV,COL+ZZH,JEU(NH+ZZV,NV+ZZH))
1100 NEXT I
1110 GOTO 540
1120 REM ******************
1130 REM ECRITURE D'UN MOT SUR L'ECRAN
1140 REM ******************
1150 L=LEN(MOT$)
1160 FOR U=1 TO L
1170 K=ASC(SEG$(MOT$,U,1))
1180 CALL HCHAR(Y,X+U,K)
1190 NEXT U
1200 RETURN
1210 REM ******************
1220 REM LECTURE DES COORDONNEES
1230 REM ******************
1240 MOT$="(LIGNE,COLONNE)-->
     "
1250 X=3
1260 Y=23
1270 GOSUB 1150
1280 CALL KEY(0,KY,S)
1290 IF S=0 THEN 1280
1300 IF KY=13 THEN 1410
1310 IF (KY<65)+(KY>64+C)=-1 THEN 1280
1320 CALL HCHAR(23,22,KY)
1330 CALL HCHAR(23,24,ASC(","))
1340 CALL KEY(0,KX,S)
1350 IF (KX<65)+(KX>64+C)=-1 THEN 1340
1360 CALL HCHAR(23,26,KX)
1370 LIGNE=KY-64+12-CC
1380 COL=KX-64+15-CC
1390 NH=KY-64
1400 NV=KX-64
1410 RETURN
1420 REM *****************
1430 REM ROUTINE ERREUR
1440 REM ******************
1450 MOT$="ATTENTION!!!!
     "
```

```
1460 Y=23
1470 X=3
1480 GOSUB 1150
1490 FOR I=1 TO 30
1500 CALL SOUND(50,440+3*I,4)
1510 NEXT I
1520 RETURN
1530 REM ******************
1540 REM *ROUTINE EFFACEMENT
1550 REM *******************
1560 CALL HCHAR(21,1,32,96)
1570 CALL HCHAR(1,1,32,64)
1580 CALL VCHAR(6,27,32,15)
1590 CALL HCHAR(6,24,32,7)
1600 RETURN
```

Ce programme est imparfait; si vous désirez le compléter, voici quelques suggestions :

- recopier le tableau JEU dans un fichier et créer un programme qui le relit;
- créer une routine de modification. L'utilisateur peut souhaiter modifier la position des noires ou reprendre un partie de la grille;
- créer une routine qui indique continuellement à l'utilisateur la longueur du mot qu'il peut écrire;
- remplacer les lettres qui désignent les lignes de la grille par des nombres...

Pour tout ceci, bon courage et envoyez-nous vos suggestions.

■

Bibliographie

Sophie Ehster

TI 99 à la conquête des jeux, de Pierre Willard, aux Editions Eyrolles - 116 pages.

Nous connaissons déjà, du même auteur, *La conduite du TI 99*, qui traitait de l'organisation interne de l'ordinateur. L'auteur récidive cette fois par un recueil de 14 jeux fonctionnant tous avec la console de base.

Tous ces programmes utilisent largement les possibilités graphiques et sonores de l'ordinateur et se veulent complets et complexes.
On retrouve dans cet ouvrage de grands classiques tels *STAR-TREK, AWARI, SCRABBLE*, et des programmes plus originaux qui sont classés en trois catégories : jeux interactifs, jeux d'aventure et jeux de réflexion.

Chaque programme est largement commenté et le lecteur, même s'il ne peut toujours appréhender la logique du programme, pourra s'inspirer des techniques de programmation employées. Un ouvrage soigné qui a sa place dans votre bibliothéque.

Jeux, Trucs et Comptes pour TI 99/4A, de Michel Benelfoul, aux Editions du PSI - 176 pages, 90 FF.

Cette série, que l'on retrouve adaptée à plusieurs ordinateurs, nous est proposée ici pour notre TI.

Il s'agit d'un ouvrage comportant trente programmes, tous en Basic TI, exécutables avec la console seule et traitant des sujets les plus divers. On y trouve des interludes graphiques, des jeux, des trucs, des utilitaires (tri,conversion), et même un programme d'édition des bulletins de salaire.

Chaque programme est présenté avec son organigramme et la liste des variables avec leur rôle, et leur signification est entièrement explicitée. On peut ainsi aisément "rentrer" dans le programme pour y faire ses propres modifications.

Ce livre qui s'adresse aux débutants vous fera progresser en Basic TI; c'est un peu sous cet aspect qu'il faut le considérer et non pas comme un simple recueil de programmes tous faits.

102 programmes pour TI 99/4A, de Jacques Deconchat, aux Editions du PSI - 238 pages, 110 FF.

Egalement dérivé d'une série pour divers ordinateurs, cet ouvrage vous offre 102 programmes (pas moins) écrits en majeure partie en Basic TI (la dernière partie du livre, qui en comporte 5, s'adresse aux possesseurs du Basic Etendu).

A un franc le programme, cet ouvrage balaie tous les compartiments des jeux. Tous les programmes présentés sont courts, mais néanmoins assez complets et performants.

Le but avoué de cet ouvrage est l'apprentissage du Basic TI par l'exemple et la pratique.
Le livre se compose de cinq parties débutant chacune par l'étude des nouvelles fonctions que l'on va rencontrer. Un texte comprenant les références du jeu, ses principales caractéristiques, une analyse du programme, un exemple d'exécution et une description des éventuelles extensions, accompagne chaque listing.

Les 102 programmes que contient ce livre devraient vous inspirer pour créer vos propres jeux.

Notons enfin que cet ouvrage, comme le précédent, a été adapté au TI 99 par notre ami Gérard Santraille.

Boîte à outils pour TI 99/4A, par Marcel Henrot, aux éditions MegaO Poche, distribué par les Editions du PSI - 128 pages, 35 FF.

Comme son nom l'indique, ce livre est un recueil de "trucs" conçus pour être utilisés avec la console, sans qu'aucune extension ne soit nécessaire.

D'une présentation originale (format de poche) il regroupe une trentaine de programmes trés courts (de 5 à 50 lignes) destinés à être insérés dans des applications plus complexes.
Certains programmes vous proposent des améliorations potentielles et vous guident pour l'élaboration de ces extensions.
La structure de chaque programme est analysée et on trouve également avant chaque listing un tableau d'identification des variables.

Un petit livre fort sympathique qui, s'il ne propose pas de programmes très élaborés ou sophistiqués, permet néanmoins de "bricoler" avec son TI 99.

■

IF... THEN... FOR...

Le Basic Etendu n'admet *théoriquement* pas les boucles "FOR-NEXT" après IF-THEN-ELSE. Si vous faites un RUN après avoir entré le programme suivant :

```
110 CALL KEY (0,K,S)
120 IF S=0 THEN 110
130 IF K=49 THEN FOR I=1 TO
10 :: PRINT I :: NEXT I :: GOTO
110 ELSE FOR I=1 TO 10 :: PRINT
"?" :: NEXT I :: GOTO 110
```

Vous obtenez le message :

```
* SYNTAX ERROR IN 130
```

Ajoutez au programme la ligne suivante :

```
100 GOTO 110 :: CALL KEY :: K ::
S :: I :: !@P−
```

Cela fonctionne ! Si nous appuyons sur la touche "1", le programme exécute une boucle qui affiche les chiffres 1 à 10. Une pression sur une autre touche provoque l'exécution de la boucle qui affiche dix points d'interrogation.

Conclusion : l'annulation du "Pré-Scan" autorise les boucles après un test.

Warangal

Edouard Krok

Ce jeu, écrit en **Basic TI**, rappelle le jeu de l'oie, mais il se joue sans dés et ne comporte pas de cases particulières. Le terrain de jeu est une spirale de 25 cases et l'on joue contre l'ordinateur.

Le premier joueur est tiré au sort lors de la première partie; ensuite, c'est le perdant de la partie précédente qui commence. Chaque joueur joue à tour de rôle une, deux ou trois marques : s'il joue une marque, le joueur suivant devra jouer sur la case suivante; s'il joue deux marques, le suivant jouera deux cases plus loin et s'il joue en trois, il faudra jouer trois cases plus loin.

Le gagnant sera celui qui parviendra à la dernière case, étant entendu que sur l'avant-dernière case on ne peut jouer que un, et sur la précédente on ne peut jouer que un ou deux.

Le TI-99/4A n'aime pas perdre lorsqu'il joue au Warangal mais, s'il vous voit en difficulté, il abaissera le niveau de son jeu. Il ne vous reste plus qu'à trouver une stratégie qui vous permette de relever honorablement le défi que vous lance votre ordinateur.

Liste des variables

GP$(5) : conserve le prénom du gagnant de chacune des 5 parties.
COM : désigne celui qui doit jouer. 0 = TI, 1 = joueur. Lors de la première partie, COM est déterminée au hasard. Ensuite, c'est celui qui vient de perdre la partie précédente qui commence.
DES : contient le numéro du graphisme qu'il faudra dessiner selon que c'est le TI ou le joueur qui joue.
K : Code ASCII du nombre de marques de jeu.
LCH : diminue lorsqu'on ne peut choisir qu'une ou deux marques de jeu.
NIV : le TI adapte le niveau de son jeu aux résultats du joueur.
PJR : points du joueur.
PTI : points du TI.
PST : indique quelle est la case où il faut jouer.
G$: gagnant de la partie.
JR$: prénom du joueur.
VQR$: vainqueurs des cinq parties.

■

```
10 REM ********************
20 REM *      Warangal    *
30 REM ********************
40 REM *      Basic TI    *
50 REM ********************
60 REM
100 DIM GP$(5)
110 RANDOMIZE
120 CALL CLEAR
130 PRINT TAB(11);"BONJOUR !": : : : : :
:"        Je m'appelle  TI": : : : :"Je s
uis ton":
140 PRINT TAB(11);"fidele ordinateur.": :
 : :"Et toi quel est ton prenom ?": : :
150 CALL SOUND(200,1000,5)
160 INPUT "             ":JR$
170 IF SEG$(JR$,2,1)>"Z" THEN 210
180 IF LEN(JR$)<13 THEN 230
190 PRINT : :"Mon programme ne me permet"
: :"d'accepter qu'un prenom": :"de 3 a
12 lettres !": :
200 GOTO 160
210 PRINT : :"Priere d'ecrire avec des":
:"LETTRES MAJUSCULES.  Merci !": :
220 GOTO 160
230 IF LEN(JR$)<3 THEN 190
240 CALL CLEAR
250 PRINT "      Bravo  ";JR$: : :"tu as b
ien fait de choisir": :"de jouer contre
 moi au": : : :
260 PRINT "      W A R A N G A L": : : :
270 PRINT "Mais attention, je suis un": :
"ADVERSAIRE REDOUTABLE !": : : :"Defend
s toi bien ";JR$;" !"
280 CALL SOUND(200,300,7)
290 FOR DELAI=1 TO 130
300 CALL KEY(3,K,S)
310 IF S<>0 THEN 430
320 NEXT DELAI
330 CALL CLEAR
340 PRINT "      W A R A N G A L":"      ~~~
~~~~~~~~~~~~~~"
350 PRINT "Le but du jeu est de": :"parve
nir a la case centrale.": : :
360 PRINT "Chaque joueur met 1,2 ou 3 ":
:"marques; et le joueur": :"suivant jou
e 1,2 ou 3 cases": :
370 PRINT "plus loin.": : :"Dans l'avant
derniere case,": :"on ne peut juoer que
 1; et": :
380 PRINT "dans celle d'avant on ne": :"p
eut jouer que 1 ou 2."
390 FOR DELAI=1 TO 350
400 CALL KEY(3,K,S)
410 IF S<>0 THEN 430
420 NEXT DELAI
430 PRINT : :
440 CALL CHAR(40,"000000FF")
450 CALL CHAR(41,"0808080808080808")
460 CALL CHAR(120,"101054FEFE7C381")
470 CALL CHAR(121,"000008082A7F7F3E")
480 CALL CHAR(122,"1C08000000000808")
490 CALL CHAR(123,"2A7F7F3E1C08")
500 CALL CHAR(130,"00000000000000FF")
510 CALL CHAR(140,"0300AA00AA0003")
520 CALL CHAR(141,"00C0300C30C")
530 CALL CHAR(150,"000000FFFFFFFFFF")
540 CALL CHAR(152,"0036777F3E1C1C08")
550 CALL CHAR(153,"0000006CEEFE7C38")
560 CALL CHAR(154,"381000000000006C")
570 CALL CHAR(155,"EEFE7C38381")
580 CALL COLOR(16,9,12)
590 CALL COLOR(15,14,1)
600 CALL COLOR(14,2,1)
```

```
610 CALL COLOR(13,7,8)
620 CALL COLOR(12,2,12)
630 CALL CLEAR
640 PRINT "      W A R A N G A L";
650 CALL COLOR(11,12,12)
660 CALL COLOR(10,5,5)
670 CALL COLOR(9,11,11)
680 CALL COLOR(4,2,1)
690 CALL COLOR(2,13,12)
700 PJR=0
710 PTI=0
720 NIV=13
730 COM=INT(2*RND)
735 REM***************************
736 REM DESSIN DU CADRE DE JEU
737 REM***************************
740 FOR L=2 TO 20
750 CALL HCHAR(L,4,112,24)
760 NEXT L
770 FOR C=3 TO 23 STEP 5
780 CALL VCHAR(2,C,41,19)
790 NEXT C
800 FOR L=5 TO 17 STEP 4
810 CALL HCHAR(L,4,40,24)
820 NEXT L
830 RESTORE 900
840 FOR I=1 TO 6
850 READ L,C,Q
860 CALL HCHAR(L,C,105,Q)
870 READ L,C,Q
880 CALL VCHAR(L,C,105,Q)
890 NEXT I
900 DATA 1,3,26,1,28,21,21,3,26,5,3,17,5,
3,21,5,23
910 DATA 13,17,8,16,9,8,9,9,8,11,9,18,5,1
3,13,6,13,13,1
920 GOTO 1010
925 REM********************************
926 REM INITIALISATION DE CAQUE PARTIE
927 REM********************************
930 RESTORE 990
940 FOR I=1 TO 24
950 READ L,C
960 CALL VCHAR(L,C,112,3)
970 CALL VCHAR(L,C+1,112,3)
980 NEXT I
990 DATA 2,5,2,10,2,15,2,20,2,25,6,25,10,
25,14,25,18,25,18,20,18,15
1000 DATA 18,10,18,5,14,5,10,5,6,5,6,10,6
,15,6,20,10,20,14,20,14,15,14,10,10,10
1010 FOR C=14 TO 17
1020 CALL VCHAR(10,C,100,3)
1030 NEXT C
1040 CALL SOUND(100,2000,6,5000,6)
1050 CALL HCHAR(11,16,63)
1055 LCH=15
1060 GOSUB 3090
1061 GOSUB 3040
1065 CALL VCHAR(1,31,130,PJR+PTI+1)
1066 GOSUB 3010
1070 PST=1
1075 REM*****************************
1076 REM COORDONNEES DE LA CASE A JOUER
1077 REM*****************************
1080 RESTORE 1120
1090 FOR I=1 TO 26-PST
1100 READ L,C
1110 NEXT I
1120 DATA 10,15,10,10,14,10,14,15,14,20,1
0,20,6,20,6,15,6,10,6,5,10,5,14,5,18,5
1130 DATA 18,10,18,15,18,20,18,25,14,25,1
0,25,6,25,2,25,2,20,2,15,2,10,2,5
1140 IF COM=1 THEN 1300
1145 REM***************************
1146 REM         JEU  TI
1147 REM***************************
1150 COM=1
1160 DES=120
1170 IF PST>22 THEN 1200
1180 IF PST<NIV THEN 1220
1190 IF PST<12 THEN 1210
1200 ON 26-PST GOTO 1280,1240,1240,1260,1
280,1220,1240,1260,1280,1220,1240,1260,
1280,1220
1210 ON 12-PST GOTO 1240,1260,1280,1220,1
240,1260,1280,1220,1240,1260,1280
1220 K=INT(3*RND)+49
1230 GOTO 1520
1240 K=49
1250 GOTO 1520
1260 K=50
1270 GOTO 1520
1280 K=51
1290 GOTO 1520
1295 REM***************************
1296 REM       REPONSE DU JOUEUR
1297 REM***************************
1300 CALL HCHAR(24,8,150,LCH+2)
1310 CALL SOUND(120,800,4)
1320 CALL SOUND(90,2000,4)
1330 COM=0
1340 DES=152
1350 FOR I=1 TO 10
1360 CALL KEY(3,K,S)
1370 IF S<>0 THEN 1410
1380 NEXT I
1390 GOSUB 3000
1400 GOTO 1350
1410 IF PST=25 THEN 1530
1420 IF K=49 THEN 1510
1430 IF PST=24 THEN 1470
1440 IF K=50 THEN 1510
1450 IF PST=23 THEN 1470
1460 IF K=51 THEN 1510
1470 CALL SOUND(200,150,0)
1480 GOSUB 3090
1490 GOSUB 3040
1500 GOTO 1300
1510 GOSUB 3100
1515 REM***************************
```

```
1516 REM        AFFICHAGE DES
1517 REM        MARQUES DE JEU
1518 REM*************************
1520 IF K=50 THEN 1570
1530 CALL SOUND(-80,1000-850*COM,6)
1540 CALL HCHAR(L+1,C+COM,DES)
1550 IF PST=25 THEN 1570
1560 IF K=49 THEN 1620
1570 CALL SOUND(-100,1200-950*COM,6)
1580 CALL HCHAR(L,C+1-COM,DES+1)
1590 CALL HCHAR(L+1,C+1-COM,DES+2)
1600 CALL SOUND(60,1800-1550*COM,6)
1610 CALL HCHAR(L+2,C+1-COM,DES+3)
1615 REM*************************
1617 REM        MISE  A  JUOR
1618 REM     POSITION ET CHOISIR
1619 REM*************************
1620 IF PST=25 THEN 1730
1630 PST=PST+K-48
1640 IF PST=24 THEN 1670
1650 IF PST=23 THEN 1690
1660 GOTO 1080
1670 LCH=11
1680 GOTO 1700
1690 LCH=13
1700 GOSUB 3090
1710 GOSUB 3040
1720 GOTO 1080
1725 REM*************************
1726 REM   FIN DE CHAQUE PARTIE
1727 REM*************************
1730 CALL HCHAR(3,1,32,2)
1740 IF COM=1 THEN 1800
1750 PJR=PJR+1
1760 GP$(PJR+PTI)=JR$
1770 NIV=NIV-6
1780 G$="PARTIE GAGNEE PAR "&JR$
1790 GOTO 1840
1800 PTI=PTI+1
1810 GP$(PJR+PTI)="TI"
1820 NIV=NIV+6
1830 G$="PARTIE GAGNEE PAR  TI"
1840 L=LEN(G$)
1850 C=(30-L)/2
1860 GOSUB 3090
1870 CALL SOUND(1500,-4,10)
1880 FOR I=1 TO L
1890 CALL HCHAR(23,C+I,ASC(SEG$(G$,I,1)))
1900 NEXT I
1910 CALL SOUND(-100,1200,4)
1920 CALL HCHAR(9,31,48+PJR)
1930 CALL SOUND(100,300,5)
1940 CALL HCHAR(15,31,48+PTI)
1950 IF PJR+PTI<5 THEN 930
1955 REM***************************
1956 REM        FIN  DE  JEU
1957 REM***************************
1960 FOR DELAI=500 TO 2500 STEP 300
1970 CALL SOUND(350,DELAI,3)
1980 NEXT DELAI
1990 CALL CLEAR
2000 FOR I=2 TO 11
2010 CALL COLOR(I,2,1)
2020 NEXT I
2030 PRINT TAB(11);"WARANGAL": : : :
2040 FOR I=1 TO 5
2050 PRINT "    partie";I;":  ";GP$(I): :
 :
2060 NEXT I
2070 IF PTI>PJR THEN 2100
2080 VQR$="Vainqueur :  "&JR$
2090 GOTO 2110
2100 VQR$="Vainqueur :  TI"
2110 PRINT :TAB((30-LEN(VQR$))/2);VQR$: :
2120 CALL SOUND(200,800,5,1000,6)
2130 FOR I=500 TO 3000 STEP 300
2140 CALL HCHAR(23,3,32,28)
2150 CALL HCHAR(23,(32-LEN(VQR$))/2,150,2
+LEN(VQR$))
2160 CALL SOUND(400,I,10,500,15,3200-I,20
,-8,20)
2170 NEXT I
2180 PRINT TAB(9);"R_Recommencer":TAB(9);
"F_Finir";
2190 CALL KEY(3,K,S)
2200 IF S=0 THEN 2190
2210 IF K=82 THEN 630
2220 IF K=70 THEN 120
2230 GOTO 2190
2995 REM****************************
2996 REM  CLIGNOTEMENT DE LA FLECHE
2997 REM****************************
3000 CALL HCHAR(3,1,32,2)
3010 CALL HCHAR(3,1,140)
3020 CALL HCHAR(3,2,141)
3030 RETURN
3035 REM**************************
3036 REM       ECRIRE :  CHOISIR
3037 REM**************************
3040 CALL SOUND(-1500,600,15,-8,10)
3050 FOR I=1 TO LCH
3060 CALL HCHAR(23,8+I,ASC(SEG$("CHOISIR
: 1 2 3",I,1)))
3070 NEXT I
3080 RETURN
3090 CALL HCHAR(23,1,32,32)
3100 CALL HCHAR(24,1,32,32)
3110 RETURN
```

Depuis l'arrêt de la fabrication du TI 99/4A, il est difficile de se procurer, hormis par la revente de particuliers, certaines extensions (boîtier, lecteurs de disquettes...).

Des sociétés françaises seraient prêtes, dit-on, à concevoir des périphériques compatibles avec le TI.

Je propose aux personnes intéressées de m'écrire, afin de recenser les besoins, qui transmis à ces sociétés feraient peut-être avancer les choses.

Mon adresse :
Marc Roch-Vilato
33, boucle de la Milliaire
57100 Thionville.

Graphisme en haute résolution (suite)

Gérard Santraille et Jean-Luc Bazanegue

Nous vous proposons de rajouter à l'ensemble de routines graphiques du dernier numéro, un sous programme baptisé *LIGNE* qui permet de tracer, en haute résolution évidemment, un segment de droite entre deux points quelconques.

```
        DEF   EFFT,EFFP,EFFC,AXES,POINT,IMAGE
        DEF   LIGNE

        AORG  >A000

COORD   DATA  0
COULP   DATA  0
*
COULE   DATA  0
POSAXE  DATA  0
LIGNED  DATA  0
LIGNEA  DATA  0
REG11   DATA  0
CR7     DATA  0
CR8     DATA  0
TBASIC  BSS   >4000
TIMAGE  BSS   >1800
RVDP1   DATA  >E181
  .       .     .
  .       .     .
  .       .     .
LIGNE   MOVB  @LIGNED,R9
        SRL   R9,8
        MOVB  @LIGNEA,R7
        SRL   R7,8
        MOV   R7,@CR7
        MOVB  @LIGNED+1,R2
        SRL   R2,8
        CI    R2,191
        JGT   LIGNE8
        LI    R10,191
        S     R2,R10
        MOVB  @LIGNEA+1,R2
        SRL   R2,8
        CI    R2,191
        JGT   LIGNE8
        LI    R8,191
        S     R2,R8
        MOV   R8,@CR8
*
        CLR   R12
        LI    R5,1
        LI    R6,1
        S     R9,R7
        JLT   LIGNE1
        JMP   LIGNE2
LIGNE1  NEG   R5
        NEG   R7
LIGNE2  MOV   R7,R7
        JNE   LIGNE3
        SETO  R12
LIGNE3  S     R10,R8
        JLT   LIGNE4
        JMP   LIGNE5
LIGNE4  NEG   R6
        NEG   R8
LIGNE5  MOV   R9,R0
        MOV   R10,R1
        MOV   R1,R4
        SLA   R4,5
        SOC   R1,R4
        ANDI  R4,>FF07
        A     R0,R4
        ANDI  R0,7
        S     R0,R4
        LI    R3,>8000
        SRC   R3,0
        SOCB  R3,@TIMAGE(R4)
        MOVB  @COULP,R13
        CLR   R2
        MOVB  @COULP+1,R2
        SRC   R2,12
        SOCB  R2,R13
        MOVB  R13,@TCOUL(R4)
        C     R9,@CR7
        JNE   LIGNE6
        C     R10,@CR8
        JNE   LIGNE6
LIGNE8  B     @RETOUR
LIGNE6  MOV   R12,R12
        JLT   LIGNE7
        A     R5,R9
        S     R8,R12
        JMP   LIGNE5
LIGNE7  A     R6,R10
        A     R7,R12
        JMP   LIGNE5

EFFT    SETO  R2
        JMP   EFFP2
          .     .
          .     .
          .     .
```

Cette routine s'utilise de la même façon que les précédentes, au moyen de *CALL LINK* aprés avoir poké en :

−24568 abscisse de départ (0 à 255)
−24567 ordonnée de départ (0 à 191)
−24566 abscisse d'arrivée (0 à 255)
−24565 ordonnée d'arrivée (0 à 191)

Si un des points extrêmes d'une ligne se trouve en dehors des intervalles précisés ci-dessus, la ligne est ignorée.

Comme pour les autres routines, la couleur du tracé dépend des valeurs pokées en −24574 (couleur "OFF") et en −24573 (couleur "ON"). Ainsi :

CALL LOAD (−24568,0,0,255,191)
CALL LINK ("LIGNE")

trace une diagonale sur l'écran graphique.

L'adjonction de cette nouvelle instruction à notre bibliothèque de commandes graphiques permet d'adapter n'importe quel programme graphique à notre TI préféré. C'est notamment le cas de l'Apple II+ ou //e, pour lequel les programmes en haute résolution se comptent par centaines.

Conversion des programmes graphiques destinés à l'Apple

Nous allons passer en revue toutes les instructions graphiques de l'Apple, et donner leur équivalent en Basic TI.

HGR active le mode graphique haute résolution (280 par 160), et laisse quatre lignes de texte au bas de l'écran (POKE −16302,0 ou POKE 49234,0 supprime cette fenêtre de texte). Cette instruction n'a pas d'équivalent; en effet, sur le TI le graphique n'est pas visible en permanence et le mode *bit-map* n'est activé que lors de la visualisation de ce dernier lors de l'exécution de CALL LINK ("IMAGE").

HGR2 active le mode graphique haute résolution sur la totalité de l'écran et affiche la deuxième page graphique. Notre TI n'ayant pas assez de mémoire pour contenir les différents tampons nécessaires au stockage de deux images graphiques, cette instruction n'a pas d'équivalent, sauf si l'on supprime le tampon *TCOUL* car, dans ce cas, la mémoire ainsi libérée pourrait être utilisée pour créer une deuxième page. Il est évident que les images ainsi générées seraient monochromes.

HCOLOR=I , avec I variant de 0 à 7, définit la couleur des tracés suivants. CALL LOAD(−24573,I) permet de simuler cette instruction. Notons au passage que la palette de couleurs du TI est plus riche puisqu'elle permet d'afficher 16 couleurs (contre 8 sur l'Apple).

HPLOT est l'instruction fondamentale du tracé en haute résolution sur Apple. Elle se présente sous deux formes différentes :

HPLOT X,Y allume le point d'abscisse X et d'ordonnée Y. Son équivalent est
CALL LOAD (−24576,255*X/280,Y)
CALL LINK ("POINT")

Notez que l'abscisse X est multipliée

par 255/280 car la résolution du TI (256x192) est légèrement plus faible que celle de l'Apple (280x192).

HPLOT X1,Y1 TO X2,Y2 trace un segment de droite entre les deux points considérés. L'équivalent de cette instruction est :
CALL LOAD (−24568,255/280*X1, Y1,255/280*X2,Y2)
CALL LINK("LIGNE")

Remarque : HPLOT peut être employée avec plus de deux points selon : HPLOT X1,Y1 TO X2,Y2 TO X3,Y3 etc... Il convient alors d'employer "LIGNE" autant de fois que nécessaire.

Programme de démonstration

Pour illustrer l'emploi de l'instruction *LIGNE*, nous vous proposons un petit programme graphique permettant de tracer un fractal, un hyperparallélépipède ou un ensemble de droites concourantes.

La technique employée pour tracer un fractal de base et d'ordre quelconques s'inspire d'un programme initialement destiné à l'Apple.

Le programme procède par menus successifs.

Tracé d'un fractal

Ne cherchez pas dans un dictionnaire le terme *FRACTAL*, il n'y figure pas encore. Il s'agit d'une entité purement abstraite, et assez difficile à appréhender dans son ensemble, et pourtant extrêmement facile à construire pas à pas.

Considérons un banal segment de droite, divisons le en trois parties égales, remplaçons celle du centre par un triangle équilatéral construit sur elle : on obtient un fractal d'ordre un.
En faisant de même avec les quatre segments ainsi obtenus, on génère un fractal d'ordre deux, et ainsi de suite...

La figure ainsi construite devient (lorsque l'ordre ou la complexité tend vers l'infini) une *courbe de Von Koch* (du nom du mathématicien qui l'imagina en 1904).

La propriété la plus étonnante de cette figure est sa dimension fractale (grandeur comparable en tout point à la dimension d'un espace affine ou vectoriel); elle vaut log(4) / log(3) = 1,26. Vous avez bien lu : 1,26. La dimension n'est pas entière. Sans s'appuyer sur de trop lourdes considérations mathématiques, on peut essayer d'appréhender cette apparente incongruité. En effet, on comprend aisément que cette dimension est bien inférieure à deux (qui est la dimension du plan entier). De la même façon, la dimension ne peut qu'être supérieure à celle d'une droite (en l'occurence un) puisqu'il s'agit d'une figure autrement plus complexe et dont le point de départ est une droite. Toute discussion sur les fractaux peut sembler ésotérique, il faut néanmoins savoir qu'ils constituent un ensemble bien quantifié et que les démonstrations qu'ils mettent en jeu sont rigoureusement exactes.

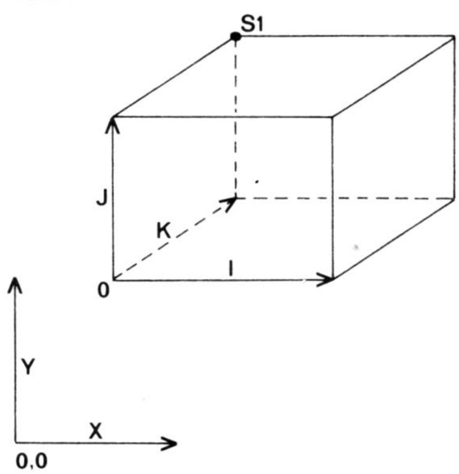
Voyons plutôt leur aspect pittoresque et la trés grande pureté de leurs lignes. Le programme vous permet de tracer un fractal d'ordre 0 à 5 (cette limitation à 5 provient de la résolution du TI) et de choisir le motif de départ .Ce dernier peut être un segment de droite (choix 1 : la fondamentale), un triangle avec une expansion vers l'intérieur ou l'extérieur (choix 2 ou 3) ou un hexagone avec, comme pour le triangle, une expansion vers l'intérieur ou l'extérieur (choix 4 et 5).
Le menu, par l'option 6, permet en outre de vérifier la réversibilité du tracé d'un fractal.

Laissez-vous griser par ces courbes trés pures, modifiez la figure de départ, superposez vos constructions...

Tracé d'un hyperparallélépipède

La deuxième option du menu principal vous permet de dessiner un hyperparallélépipède, c'est-à-dire un ensemble de parallélépipèdes imbriqués les uns dans les autres, ayant la même origine et en commun plusieurs vecteurs directeurs.

Considérons le parallélépipède suivant :



Quatre données suffisent à le définir :
- son origine (O)
- ses trois vecteurs directeurs I, J et K

Chaque sommet du parallélépipède peut alors s'écrire comme une combinaison linéaire de I, J et K, où les coefficients vaudraient 0 ou 1.

exemple : S1 -> OS1 = 0 x I + 1 x J + 1 x K. Chacun des trois vecteurs directeurs du parallélépipède peut s'écrire à son tour en fonction des deux vecteurs X et Y, qui définissent le plan de l'écran de notre TI. Rappelons à ce sujet que le plan graphique a son origine (0,0) en bas et à gauche de l'écran, que l'axe X est celui des abscisses et Y celui des ordonnées.

Le programme, lorsqu'on choisit de dessiner un hyperparallélépipède, demande où il doit placer l'origine, puis le nombre de vecteurs directeurs (3 correspond à un parallélépipède classique) et enfin permet de rentrer pour chaque vecteur ses deux composantes.

Voici un exemple de valeurs que vous pouvez entrer pour créer un hyperpallélépipède :
Origine : 80,140
Dimensions : 5
V1 : 70,0
V2 : 0,70
V3 : 40,−40
V4 : −50,30
V5 : 40,25

Tracé de 200 droites concourantes de couleurs aléatoires

Le choix de cette option provoque l'affichage d'un ensemble de droites concourantes. L'analyse du programme Basic se passe de commentaires; à noter simplement que l'image obtenue rappelle un peu le programme LINES de la mini-mémoire.

```
100 REM******************
110 REM* DEMONSTRATION  *
120 REM*   GRAPHIQUE    *
130 REM******************
140 REM*TI-BASIC et Ed/As*
150 REM******************
160 REM*   Copyright    *
170 REM* "99 Magazine"  *
180 REM*Gerard Santraille*
190 REM*   et Jean-Luc  *
200 REM*   Bazaneque    *
210 REM******************
220 REM
230 CALL CLEAR
240 CALL SCREEN(8)
250 PRINT ,"Chargement.."
260 CALL INIT
270 CALL LOAD("DSK1.OBJETL")
280 CALL CHAR(95,"0000FFFF")
290 CALL CLEAR
300 PRINT "PROGRAMMES DE DEMONST
RATION":"________________
____": : :"1-Tra
ce d'un fractal"
310 PRINT :"2-Trace d'un hyper-"
:" parallelepipede"
320 PRINT :"3-Droites concourant
es":" de couleurs aleatoires":
```

```
:"4-Finir": : :,
"Votre choix :": :
330 L=22
340 H=31
350 GOSUB 1820
360 IF (V<49)+(V>52)THEN 350
370 IF V-52 THEN 400
380 CALL CLEAR
390 STOP
400 CALL LOAD(-24574,0,11,12)
410 CALL CLEAR
420 ON V-48 GOSUB 440,1310,1650
430 GOTO 290
440 PRINT "TRACE D'UN FRACTAL":"
________________"
450 U1=1/3
460 U2=1/2
470 V2=SQR(3)/6
480 U3=2/3
490 P6=3.1415926/6
500 PRINT : : : :"1-La fondament
ale":"2-Motifs dans un triangle"
:"3-Motifs hors
d'un triangle"
510 PRINT "4-Motifs dans un hexa
gone":"5-Motifs hors d'un hexago
ne":"6-Verifier
la reversibilite": : :
520 PRINT "Votre choix :":"Compl
exite (0 a 5) :": :
530 L=21
540 H=17
550 GOSUB 1820
560 K=V-48
570 IF (K>6)+(K<1)THEN 530
580 L=22
590 H=24
600 GOSUB 1820
610 C=V-48
620 CALL LINK("EFFT")
630 ON K GOSUB 670,740,830,920,1
010,1100
640 GOSUB 1910
650 CALL LINK("IMAGE")
660 RETURN
670 X1(C)=10
680 Y1(C)=150
690 X2(C)=250
700 Y2(C)=150
710 GOSUB 1990
720 C=C-1
730 RETURN
740 FOR J=-5 TO 3 STEP 4
750 X1(C)=130+100*COS(J*P6)
760 Y1(C)=109-100*SIN(J*P6)
770 X2(C)=130+100*COS((J+4)*P6)
780 Y2(C)=109-100*SIN((J+4)*P6)
790 GOSUB 1990
800 C=C-1
810 NEXT J
820 RETURN
830 FOR J=11 TO 3 STEP -4
840 X1(C)=130+86*COS(J*P6)
850 Y1(C)=96-86*SIN(J*P6)
860 X2(C)=130+86*COS((J-4)*P6)
870 Y2(C)=96-86*SIN((J-4)*P6)
880 GOSUB 1990
890 C=C-1
900 NEXT J
910 RETURN
920 FOR J=-1 TO 9 STEP 2
930 X1(C)=130+86*COS(J*P6)
940 Y1(C)=96-86*SIN(J*P6)
950 X2(C)=130+86*COS((J+2)*P6)
960 Y2(C)=96-86*SIN((J+2)*P6)
970 GOSUB 1990
980 C=C-1
990 NEXT J
1000 RETURN
1010 FOR J=11 TO 1 STEP -2
1020 X1(C)=130+80*COS(J*P6)
1030 Y1(C)=96-80*SIN(J*P6)
1040 X2(C)=130+80*COS((J-2)*P6)
1050 Y2(C)=96-80*SIN((J-2)*P6)
1060 GOSUB 1990
1070 C=C-1
1080 NEXT J
1090 RETURN
1100 X1(C)=10
1110 Y1(C)=150
1120 X2(C)=250
1130 Y2(C)=150
1140 GOSUB 1990
1150 C=C-1
1160 IF C THEN 1180
1170 RETURN
1180 C=C-1
1190 X1(C)=250
1200 Y1(C)=150
1210 X2(C)=130
1220 Y2(C)=75
1230 GOSUB 1990
1240 C=C-1
1250 X1(C)=130
1260 Y1(C)=75
1270 X2(C)=10
1280 Y2(C)=150
1290 GOSUB 1990
1300 RETURN
1310 PRINT "TRACE D'HYPERPARALLE
LEPIPEDE":"____________________
_______": : : :
1320 DIM X(100),Y(100)
1330 INPUT "Origine (X,Y) : ":X0
,Y0
1340 INPUT "Dimension : ":K
1350 CALL LINK("EFFT")
1360 FOR I=0 TO K-1
1370 INPUT "Vecteur (VX,VY) : ":
VX(I),VY(I)
1380 B(I)=0
1390 NEXT I
1400 PRINT :"Trace en cours"
1410 FOR I=0 TO 2^K-1
1420 N=I
1430 FOR J=K-1 TO 0 STEP -1
1440 B(J)=INT(N/2^J)
1450 N=N-B(J)*2^J
1460 NEXT J
1470 X(I)=X0
1480 Y(I)=Y0
1490 FOR J=0 TO K-1
1500 IF B(J)=0 THEN 1530
1510 X(I)=X(I)+VX(J)
1520 Y(I)=Y(I)-VY(J)
1530 NEXT J
1540 FOR J=0 TO K-1
1550 IF B(J)<>0 THEN 1590
1560 XV(J)=X(I)+VX(J)
1570 YV(J)=Y(I)-VY(J)
1580 CALL LOAD(-24568,X(I),Y(I),
XV(J),YV(J))
1590 CALL LINK("LIGNE")
1600 NEXT J
1610 NEXT I
1620 GOSUB 1910
1630 CALL LINK("IMAGE")
1640 RETURN
1650 PRINT "DROITES CONCOURANTES
 DE":"__________________________":"
COULEURS ALEATOI
RES.."
1660 PRINT "____________________
_": : : : : :"Trace en cours"
1670 CALL LINK("EFFT")
1680 RANDOMIZE
1690 CALL LOAD(-24568,INT(RND*15
0)+50,INT(RND*100)+50)
1700 CALL LOAD(-24574,0,0,1)
1710 FOR I=0 TO 200
1720 RANDOMIZE
1730 CALL LOAD(-24573,INT(RND*13
)+3)
1740 CALL LOAD(-24566,INT(RND*25
5)+1,INT(RND*191)+1)
1750 CALL LINK("LIGNE")
1760 NEXT I
1770 GOSUB 1910
1780 CALL LINK("IMAGE")
1790 RETURN
1800 REM  SP saisie choix
1810 REM  **************
1820 CALL SOUND(100,700,0)
1830 CALL HCHAR(L,H,30)
1840 CALL KEY(5,V,S)
1850 CALL HCHAR(L,H,32)
1860 IF S=0 THEN 1830
```

```
1870 CALL HCHAR(L,H,V)
1880 RETURN
1890 REM SP trace fini
1900 REM ************
1910 FOR I=1 TO 4 STEP .5
1920 CALL SOUND(40,300*I,0)
1930 NEXT I
1940 FOR I=1 TO 50
1950 NEXT I
1960 RETURN
1970 REM SP dessin recursif
1980 REM *****************
1990 IF C THEN 2040
2000 CALL LOAD(-24568,X1(C),Y1(C
),X2(C),Y2(C))
2010 CALL LINK("LIGNE")
2020 C=C+1
2030 RETURN
2040 DX(C)=X2(C)-X1(C)
2050 DY(C)=Y2(C)-Y1(C)
2060 X1(C-1)=X1(C)
2070 Y1(C-1)=Y1(C)
2080 X2(C-1)=X1(C)+U1*DX(C)
2090 Y2(C-1)=Y1(C)+U1*DY(C)
2100 C=C-1
2110 GOSUB 1990
2120 X1(C-1)=X2(C-1)
2130 Y1(C-1)=Y2(C-1)
2140 X2(C-1)=X1(C)+U2*DX(C)+V2*D
Y(C)
2150 Y2(C-1)=Y1(C)+U2*DY(C)-V2*D
X(C)
2160 C=C-1
2170 GOSUB 1990
2180 X1(C-1)=X2(C-1)
2190 Y1(C-1)=Y2(C-1)
2200 X2(C-1)=X1(C)+U3*DX(C)
2210 Y2(C-1)=Y1(C)+U3*DY(C)
2220 C=C-1
2230 GOSUB 1990
2240 X1(C-1)=X2(C-1)
2250 Y1(C-1)=Y2(C-1)
2260 X2(C-1)=X2(C)
2270 Y2(C-1)=Y2(C)
2280 C=C-1
2290 GOSUB 1990
2300 C=C+1
2310 RETURN
```

Ce petit programme de démonstration vous montre comment, avec peu d'instructions, on peut créer de très beaux graphiques. Il est désormais facile de manier la haute résolution du TI; ne vous en privez pas....

Pour faire suite à la demande de nombreux lecteurs, nous avons constitué une disquette contenant le fichier source modifié (ajout de la routine *LIGNE*), un code objet utilisable avec la Mini-mémoire et un code objet prévu pour l'Editeur/Assembleur.

Vous pouvez acquérir cette disquette en nous envoyant le bon de commande joint à ce numéro. ■

Le Trésor de l'Armada

Georges Herbepin

N.D.L.R. : si un lecteur de disquettes est raccordé à votre TI, vous devez utiliser le désormais célèbre "CALL FILES(1)" avant de charger ce programme écrit en **Basic Etendu**.

La recherche de trésors submergés occupe depuis longtemps de nombreux plongeurs amateurs ou professionnels. Ce programme vous propose de prendre leur place sans pour autant vous mouiller.

Le "mode d'emploi" du jeu s'affiche en début d'exécution du programme, vous pouvez donc vous jeter à l'eau dès à présent. ■

```
10 ! ************************
20 ! *    Le tresor de      *
30 ! *      l'armada        *
40 ! ************************
50 ! *    Basic Etendu      *
60 ! ************************
70 !
100 CALL CLEAR :: FOR A=1 TO 8 :: CALL CO
LOR(A,12,5):: NEXT A :: CALL SCREEN(5)
110 DISPLAY AT(1,3):"* LE TRESOR DE L'ARM
ADA *" :: DISPLAY AT(3,3):"*************
*************"
120 X=2^(1/20):: FOR A=1 TO 22 :: CALL SO
UND(80,110*X^A,0):: NEXT A
130 DISPLAY AT(5,3):"UN TRESOR(COFFRE REN
FER-":"MANT DES PIECES D'OR)REPOSE":"AU
 FOND DE LA MER,GARDE PAR"
140 DISPLAY AT(8,1):"UNE REDOUTABLE PIEUV
RE.VOUS":"DISPOSEZ DE 5 PLONGEURS DONT"
:"LES DEPLACEMENTS SONT CON-"
150 DISPLAY AT(11,1):"TROLES PAR LA MANET
TE NO 1":"ET LE DEPART PAR LE BOUTON":"
DE TIR.DES QU'UN COFFRE EST"
160 DISPLAY AT(14,1):"RAMENE SUR LE BATEA
U,UN NOU-":"VEAU TRESOR APPARAIT ALEA-"
:"TOIREMENT.LE PLONGEUR PEUT"
170 DISPLAY AT(17,1):"ETRE DETRUIT PAR LA
 PIEUVRE":"ET,UNIQUEMENT S'IL A ATTRA-"
:"PE LE TRESOR,PAR LA MEDUSE"
180 DISPLAY AT(20,1):"OU L'HUITRE SI ELLE
 EST":"OUVERTE."
190 DISPLAY AT(23,1):"APPUYEZ SUR UNE TOU
CHE":"POUR LIRE LA SUITE."
200 CALL KEY(0,K,S):: IF S=0 THEN 200 ELS
E CALL CLEAR
210 DISPLAY AT(3,3):"L'HUITRE OUVERTE PEU
T RE-":"TARDER LA PIEUVRE.LE PLON-":"GE
UR DOIT OBLIGATOIREMENT"
220 DISPLAY AT(6,1):"FAIRE 2 PALIERS EN R
EMON-":"TANT AVEC LE TRESOR.UN TRE-":"S
OR ATTRAPE RAPPORTE 200"
230 DISPLAY AT(9,1):"PIECES D'OR;UN TRESO
R RA-":"MENE 1000;LA PIEUVRE RE-":"TARD
EE 50.CHAQUE PALIER ET"
240 DISPLAY AT(12,1):"CHAQUE PASSAGE PRES
 DES EPA-":"VES ENLEVE 10 PIECES D'OR."
250 DISPLAY AT(14,3):"UN SCORE SUPERIEUR
A 8000":"VOUS REDONNE 3 PLONGEURS."
260 DISPLAY AT(17,1):"RELEVEZ LA TOUCHE A
LPHALOCK.":"VOTRE SCORE S'AFFICHE LORS-
":"QUE TOUS LES PLONGEURS"
270 DISPLAY AT(20,1):"SONT MORTS."
280 DISPLAY AT(23,1):"APPUYEZ SUR UNE TOU
CHE":"POUR DEBUTER LE JEU."
290 CALL KEY(0,K,S):: IF S=0 THEN 290
300 CALL CLEAR :: CALL SCREEN(5):: FOR A=
1 TO 8 :: CALL COLOR(A,5,1):: NEXT A
310 CALL CHAR(32,"010F1F33337F1F0F0F1F162
625484884F0FCFEFFFFFFFE3C0000C0300C8241
21")
320 CALL CHAR(36,"0F3F7FFFFFFF7F3C0000030
C3041828480F0F8CCCCFEF8F0F0F86864A41212
21")
330 CALL CHAR(40,"0101000305050501030000
00000000C0C080E0D0D0504060")
340 CALL CHAR(44,"03010101010304050500000
0000000000604040C0C0E090D0D0")
350 CALL CHAR(48,"000000000000000070703030
000000000000000000000F0F0E0E0")
360 CALL CHAR(52,"03070F1D190F0404080931 2
020101121C0E0F0B898F0A0A0B80808C4444888
04")
370 CALL CHAR(56,"02020A0A0A1A1A1A3A3A7AF
A02FF6E3F008080C0C0E0E0F0F8FCFEFF00FFFF
FE")
380 CALL CHAR(60,"FFFFFFFFFFFFFFFF0103070
F1F3F7FFF80C0E0F0F8FCFEFF000041221C08")
:: CALL COLOR(4,16,8)
390 CALL CHAR(64,"010F3F377E7C78F0E0C0C0E
AFF3F0F03E0C08000000000000000000AAFF3EF8
C0")
400 CALL CHAR(68,"000000000000030F3F7FFFE
AFF3F0F0300000000000000C0F0FCDEFFAAFF3EF8
C0")
410 CALL CHAR(72,"00000307273E7C9F1F0B030
0000000000000000C0F8FC3EBBFAE08080")
420 CALL CHAR(76,"0103070F1F3F7FFF80C0E0F
0F8FCFEFFFFFFFFFFFFFFFFFF00"):: CALL CO
LOR(6,3,16)
430 CALL CHAR(80,"FFFFFFFFFFFFFFFFFFBEF7F
FDFFBDFFB0000000000000000000"):: CALL CO
LOR(7,5,16)
440 CALL CHAR(88,"FFFFFFFFFFFFFFFF80C0E0F
0F8FCFEFF0103070F1F3F7FFFFFFFEFCF8F0E0C0
80"):: CALL COLOR(8,6,5)
450 CALL CHAR(92,"FF7F3F1F0F070301FFFFDBA
5BDFFFFFF")
460 CALL CHAR(96,"432418B15A141038"):: CA
LL COLOR(9,13,5)
470 CALL CHAR(104,"FFFFFFFFFFFFFFFFFEFCF8
F0E0C08000FF7F3F1F0F070301D60C58D0C0C08
000")
480 CALL CHAR(108,"D761341707070301EFC783
C7EFEFEFC70002423409 6A9C0000"):: CALL C
OLOR(10,8,3)
490 CALL CHAR(112,"00008060301C8E673F6F83
0000000000000020C387078E8FCC080")
500 CALL CHAR(116,"000000000000080673F6F83
0F1C30400000000000000038E8FCE0F0B81C0601"
)
510 CALL CHAR(120,"00080404621A0503008890
90A3ACD0E07F030D3204040808FFE0D8A691908
888"):: CALL COLOR(12,12,8)
520 CALL CHAR(128,"000000000020201000000 0
000000000300000101010709090000806010008
080")
530 CALL CHAR(132,"10080807040C32410C30C0
00000000011121214141818101404040202011 1
608"):: CALL COLOR(13,14,5)
540 CALL CHAR(136,"40808000000000207804024
1C1C3E7FFF0101091D0F3F7FFF00000000 2070F
8DC")
550 CALL CHAR(140,"0F1F3F7FDFFF0701FFFEE4
C0E0E0C0801F0F1C0E07030100EEF7FBFEFCF8F
0C0"):: CALL COLOR(14,14,5)
```

```
560 CALL HCHAR(1,1,104,256):: CALL HCHAR(
9,1,80,512):: CALL HCHAR(8,16,78,17)::
CALL HCHAR(7,19,78,14)
570 CALL HCHAR(6,26,78,6):: CALL HCHAR(8,
15,105):: CALL HCHAR(7,18,107):: CALL H
CHAR(6,25,107):: CALL HCHAR(5,26,107):: C
ALL HCHAR(5,26,107)
580 CALL HCHAR(5,31,106):: CALL HCHAR(6,3
2,108):: CALL HCHAR(8,17,110,2):: CALL
HCHAR(8,21,110,3):: CALL HCHAR(8,25,111,2
)
590 CALL HCHAR(8,29,110,3):: CALL HCHAR(6
,27,110,3):: CALL HCHAR(5,27,77):: CALL
 HCHAR(5,29,77):: CALL HCHAR(6,20,109,4)
600 CALL HCHAR(5,28,76):: CALL HCHAR(5,30
,76):: CALL HCHAR(4,28,60,2):: CALL HCH
AR(4,27,61):: CALL HCHAR(3,28,61):: CALL
HCHAR(5,31,108)
610 CALL HCHAR(3,29,62):: CALL HCHAR(4,30
,62):: CALL VCHAR(13,2,88,12):: CALL VC
HAR(12,3,88,12):: CALL VCHAR(12,4,88,8)
620 CALL VCHAR(13,5,88,7):: CALL HCHAR(17
,6,88,5):: CALL HCHAR(18,6,88,6):: CALL
 HCHAR(19,10,88,2):: CALL HCHAR(20,11,88)
630 CALL HCHAR(21,12,88,2):: CALL HCHAR(2
2,12,88,2):: CALL HCHAR(23,10,88,3):: C
ALL HCHAR(24,9,88,4):: CALL HCHAR(16,20,8
8,2)
640 CALL HCHAR(17,16,88,7):: CALL HCHAR(1
8,15,88,8):: CALL HCHAR(19,15,88,2):: C
ALL HCHAR(19,20,88,5):: CALL HCHAR(20,21,
88,5)
650 CALL HCHAR(21,21,88,5):: CALL HCHAR(2
2,21,88,5):: CALL HCHAR(23,22,88,5):: C
ALL HCHAR(24,23,88,10):: CALL HCHAR(23,31
,88,2)
660 CALL HCHAR(22,32,88):: CALL HCHAR(12,
5,89):: CALL HCHAR(16,6,89):: CALL HCHA
R(17,11,89):: CALL HCHAR(20,12,89)
670 CALL HCHAR(16,22,89):: CALL HCHAR(18,
23,89):: CALL HCHAR(19,25,89):: CALL HC
HAR(22,26,89):: CALL HCHAR(23,27,89)
680 CALL HCHAR(12,2,90):: CALL HCHAR(20,1
4,90):: CALL HCHAR(17,15,90):: CALL HCH
AR(16,19,90):: CALL HCHAR(23,30,90)
690 CALL HCHAR(22,31,90):: CALL HCHAR(21,
32,90):: CALL HCHAR(24,3,91):: CALL HCH
AR(20,4,91):: CALL HCHAR(19,6,91)
700 CALL HCHAR(23,9,90):: CALL HCHAR(22,1
1,90):: CALL HCHAR(24,13,89):: CALL HCH
AR(23,13,91):: CALL HCHAR(21,14,91)
710 CALL HCHAR(20,15,91):: CALL HCHAR(19,
17,91):: CALL HCHAR(19,9,92):: CALL HCH
AR(20,10,92):: CALL HCHAR(21,11,92)
720 CALL HCHAR(19,19,92):: CALL HCHAR(20,
20,92):: CALL HCHAR(23,21,92):: CALL HC
HAR(24,22,92):: CALL HCHAR(11,3,96,2)
730 CALL HCHAR(16,7,96,4):: CALL HCHAR(20
,13,96):: CALL HCHAR(16,16,96,3):: CALL
 HCHAR(15,20,96,2):: CALL HCHAR(18,24,96)
740 CALL HCHAR(23,28,96,2):: CALL HCHAR(2
4,4,96,5):: CALL HCHAR(24,14,96,8):: CA
LL HCHAR(3,6,120):: CALL HCHAR(3,7,121)
750 CALL HCHAR(4,6,122):: CALL HCHAR(4,7,
123):: CALL HCHAR(12,26,128):: CALL HCH
AR(12,27,129):: CALL HCHAR(12,28,130)
760 CALL HCHAR(12,29,131):: CALL HCHAR(13
,26,132):: CALL HCHAR(13,27,133):: CALL
 HCHAR(13,28,134):: CALL HCHAR(13,29,135)
770 CALL HCHAR(14,26,136):: CALL HCHAR(14
,27,137):: CALL HCHAR(14,28,138):: CALL
 HCHAR(14,29,139):: CALL HCHAR(15,26,140)
780 CALL HCHAR(15,27,141):: CALL HCHAR(15
,28,142):: CALL HCHAR(15,29,143):: CALL
 HCHAR(21,4,128):: CALL HCHAR(21,5,129)
790 CALL HCHAR(21,6,130):: CALL HCHAR(21,
7,131):: CALL HCHAR(22,4,132):: CALL HC
HAR(22,5,133):: CALL HCHAR(22,6,134)
800 CALL HCHAR(22,7,135):: CALL HCHAR(23,
4,136):: CALL HCHAR(23,5,137):: CALL HC
HAR(23,6,138):: CALL HCHAR(23,7,139)
810 CALL HCHAR(24,4,140):: CALL HCHAR(24,
5,141):: CALL HCHAR(24,6,142):: CALL HC
HAR(24,7,143):: CALL HCHAR(10,9,81)
820 CALL HCHAR(14,13,81):: CALL HCHAR(12,
18,81):: CALL HCHAR(11,23,81):: CALL HC
HAR(18,30,81):: CALL HCHAR(21,18,81):: CA
LL HCHAR(21,9,81)
830 CALL HCHAR(2,4,63):: CALL HCHAR(3,10,
63):: CALL HCHAR(6,13,63):: CALL HCHAR(
4,21,63)
840 CALL HCHAR(14,3,93):: CALL HCHAR(18,4
,93):: CALL HCHAR(18,10,93):: CALL HCHA
R(24,11,93)
850 CALL HCHAR(17,21,93):: CALL HCHAR(20,
23,93):: CALL HCHAR(23,24,93):: CALL HC
HAR(22,32,93)
860 CALL SCREEN(2):: CALL MAGNIFY(4):: CA
LL SPRITE(#2,48,12,170,128,#3,52,15,130
,230,0,-5):: CALL SPRITE(#6,56,16,35,220,
0,-3)
870 CALL SPRITE(#7,112,5,10,4,0,12,#8,72,
15,18,220,0,-2):: CALL SPRITE(#4,64,3,9
6,45):: B=64 :: D=112
880 CALL SPRITE(#1,36,10,160,125):: P=32
:: C=36 :: NP=5 :: F=0 :: AE=0 :: Z=0 :
: AC=0 :: AP=0 :: PT=0 :: AM=0 :: BP=0
890 CALL POSITION(#6,I,G)
900 IF G>6 AND G<240 THEN 910 ELSE 890
910 CALL SPRITE(#5,40,2,49,G,0,-3):: R=40
 :: Q=5 :: H=0
920 AE=AE+1 :: IF AE=20 THEN GOSUB 1370
930 IF D=116 THEN CALL PATTERN(#7,112)::
D=112 :: GOTO 950
940 CALL PATTERN(#7,116):: D=116
950 CALL KEY(1,V,W):: IF V<>18 THEN 920
960 CALL MOTION(#Q,-15,0):: FOR A=1 TO 20
 :: NEXT A :: CALL PATTERN(#Q,44):: R=4
4
970 CALL MOTION(#Q,15,0):: FOR A=1 TO 50
:: NEXT A :: CALL SOUND(200,-2,0)
980 CALL JOYST(1,X,Y):: CALL POSITION(#Q,
S,T)
990 IF Z>=8 THEN 1010
1000 IF H=2 AND Y>=0 THEN 1390 ELSE 1020
1010 AC=AC+1 :: IF AC<=3 THEN 1020 ELSE Z
=0 :: AC=0
1020 CALL POSITION(#Q,S,T)
1030 IF S<60 OR S>170 THEN GOSUB 1460
1040 IF T<14 OR T>234 THEN GOSUB 1510
1050 IF Y>=0 THEN 1070
1060 IF R=44 THEN 1080 ELSE CALL PATTERN(
#Q,44):: R=44 :: GOTO 1080
1070 IF R=40 THEN 1080 ELSE CALL PATTERN(
#Q,40):: R=40
1080 IF H=0 THEN 1100
1090 CALL MOTION(#2,-Y,X)
1100 CALL MOTION(#Q,-Y,X)
1110 F=F+1 :: IF F=6 THEN GOSUB 1560
1120 IF D=112 THEN CALL PATTERN(#7,116)::
 D=116 :: GOTO 1140
1130 CALL PATTERN(#7,112):: D=112
1140 CALL POSITION(#1,M,N):: CALL POSITIO
N(#Q,S,T)
1150 IF N<=T THEN 1170
1160 IF C=32 THEN 1180 ELSE CALL PATTERN(
#1,32):: C=32 :: GOTO 1180
1170 IF C=36 THEN 1180 ELSE CALL PATTERN(
#1,36):: C=36
1180 IF N>=T AND M<S THEN K=-3 :: L=-3 ::
 GOTO 1220
1190 IF N<T AND M<S THEN K=3 :: L=-3 :: G
OTO 1220
1200 IF N<T AND M>=S THEN K=3 :: L=3 :: G
OTO 1220
1210 IF N>=T AND M>=S THEN K=-3 :: L=3
1220 IF AP=1 THEN 1240
1230 CALL MOTION(#1,-L,K):: GOTO 1260
1240 IF B=64 THEN CALL MOTION(#1,0,0):: P
T=PT+50 :: GOTO 1260
1250 AP=0 :: GOTO 1230
1260 CALL COINC(#1,#Q,4,AR)
1270 IF AR<>0 THEN 1580
1280 IF H=0 THEN GOSUB 1710
1290 IF H=1 THEN 1110
1300 IF H=2 THEN GOSUB 1740
1310 IF B=64 AND H=2 THEN GOSUB 1770
1320 CALL COINC(#Q,90,200,20,AX):: IF AX<
>0 THEN GOSUB 1810
1330 CALL COINC(#Q,176,30,20,AV):: IF AV<
>0 THEN GOSUB 1810
1340 IF BP=1 THEN 980
1350 IF PT>=8000 THEN NP=NP+3 :: BP=1 ::
GOTO 980
1360 GOTO 980
1370 IF P=32 THEN CALL PATTERN(#1,P):: P=
36 :: C=32 :: AE=0 :: RETURN
1380 CALL PATTERN(#1,P):: P=32 :: C=36 ::
 AE=0 :: RETURN
1390 IF S>=130 AND S<=145 THEN 1420
1400 IF S>=75 AND S<=90 THEN 1420
1410 GOTO 1020
1420 CALL POSITION(#Q,S,T):: IF T<14 OR T
>234 THEN 1440
1430 Z=Z+1 :: Y=0 :: PT=PT-10 :: GOTO 102
0
1440 IF T<14 THEN CALL LOCATE(#2,S,14)::
CALL LOCATE(#Q,S,14):: GOTO 1430
1450 CALL LOCATE(#2,S,234):: CALL LOCATE(
#Q,S,234):: GOTO 1430
1460 IF S<60 THEN 1490
1470 IF H=2 THEN CALL LOCATE(#2,170,T)
1480 CALL LOCATE(#Q,170,T):: RETURN
1490 IF H=2 THEN CALL LOCATE(#2,60,T)
1500 CALL LOCATE(#Q,60,T):: RETURN
1510 IF T<14 THEN 1540
1520 IF H=2 THEN CALL LOCATE(#2,S,234)
1530 CALL LOCATE(#Q,S,234):: RETURN
1540 IF H=2 THEN CALL LOCATE(#2,S,14)
1550 CALL LOCATE(#Q,S,14)
1560 IF B=64 THEN CALL PATTERN(#4,68):: B
=68 :: F=0 :: RETURN
1570 CALL PATTERN(#4,64):: B=64 :: F=0 ::
 RETURN
1580 IF Q=2 THEN CALL MOTION(#1,0,0):: CA
LL POSITION(#2,BE,BF):: CALL LOCATE(#1,
160,BF):: GOTO 890
1590 NP=NP-1 :: CALL SOUND(400,-6,0):: CA
LL MOTION(#1,0,0):: CALL DELSPRITE(#Q):
: Q=2
1600 IF NP=0 THEN 1820
1610 IF H=0 THEN 1650
1620 CALL MOTION(#2,8,0)
1630 CALL POSITION(#2,AF,AG):: IF AF<169
THEN 1630
1640 CALL MOTION(#2,0,0)
1650 H=1 :: AP=0 :: GOTO 1110
1660 CALL POSITION(#6,AU,AV):: CALL LOCAT
E(#Q,49,AV):: CALL LOCATE(#2,49,AV):: C
ALL MOTION(#Q,0,-3)
1670 CALL MOTION(#2,0,-3):: CALL MOTION(#
1,12,0):: CALL SOUND(200,220,0)
1680 CALL POSITION(#1,M,N):: IF M<160 THE
N 1680
1690 CALL MOTION(#1,0,0):: RANDOMIZE :: B
C=INT(RND*224)+6 :: CALL DELSPRITE(#2):
: CALL DELSPRITE(#Q)
1700 CALL SPRITE(#2,48,12,170,BC):: Q=2 :
: PT=PT+1000 :: H=1 :: GOTO 1110
1710 CALL COINC(#2,#Q,8,U):: IF U=0 THEN
RETURN
1720 CALL MOTION(#Q,0,0):: CALL POSITION(
#2,SA,TA):: CALL LOCATE(#Q,SA,TA)
1730 CALL SOUND(50,110,0):: H=2 :: PT=PT+
200 :: RETURN
1740 CALL COINC(#Q,#6,27,AF):: IF AF<>0 T
HEN 1660
1750 CALL COINC(#Q,#3,12,AH):: IF AH<>0 T
HEN 1580
1760 RETURN
1770 CALL COINC(#Q,#4,12,AJ):: IF AJ<>0 T
HEN CALL SOUND(50,330,0):: CALL PATTERN
(#4,68):: B=68 :: GOTO 1580
1780 IF AP=1 THEN RETURN ELSE CALL COINC(
#1,#4,15,AK)
1790 IF AK<>0 THEN AP=1 :: CALL SOUND(80,
440,0):: F=0 :: RETURN
1800 RETURN
1810 CALL SOUND(50,550,0):: PT=PT-10 :: R
ETURN
1820 CALL DELSPRITE(ALL):: X=2^(1/20):: F
OR A=22 TO 1 STEP -1
1830 CALL SOUND(60,110*X^A,0):: NEXT A ::
 CALL CLEAR :: CALL CHARSET :: FOR A=1
TO 8 :: CALL COLOR(A,12,2):: NEXT A
1840 DISPLAY AT(5,8)BEEP:"TOUS VOS PLONGE
URS":"SONT MORTS...!!!"
1850 DISPLAY AT(8,8):"*********" :: DISP
LAY AT(12,1):"VOTRE TRESOR SE MONTE A:"
 :: IF PT<0 THEN PT=0
1860 DISPLAY AT(14,15):PT :: DISPLAY AT(1
6,8):"PIECES D'OR..!" :: FOR A=1 TO 500
 :: NEXT A
1870 DISPLAY AT(20,1)BEEP:"APPUYEZ SUR UN
E TOUCHE":"POUR REJOUER."
1880 CALL KEY(0,K,S):: IF S=0 THEN 1880 E
LSE 300
```

Ecriture littérale d'un nombre

Christine Julien

Suite au concours que vous proposiez dans le numéro 4, je vous envoie ce petit programme en **Pascal** qui écrit tout entier positif inférieur à un million. On peut aisément le modifier pour qu'il accepte des entiers plus grands; il suffit pour cela de changer le nombre de "digits" du type entier et de rajouter une procédure million.

Ce programme respecte les règles d'orthographe relatives à l'écriture des nombres, à savoir :

- **mille** est toujours invariable (il ne prend jamais de "s");
- **cent** et **vingt** prennent un "s" que s'ils terminent l'adjectif numéral (deux cents, deux cent trente-trois);
- **le trait d'union** est obligatoire entre deux éléments s'ils sont tous deux inférieurs à cent et s'ils ne sont pas déjà reliés par un "et".

Le programme que je vous présente illustre assez bien ce que l'on appelle, en Pascal, l'analyse descendante.
En effet, une fois le nombre saisi, le programme regarde s'il est plus grand que mille, agit en conséquence puis regarde si le reste est plus grand que cent, etc... Ce découpage du traitement en procédures facilite grandement la résolution du problème car il colle d'assez près à l'analyse naturelle que l'on est amené à faire. On perçoit là l'un des nombreux avantages du Pascal.

Points particuliers du programme

Le (*$R-*) de la première ligne est une directive de compilation indiquant au programme qu'il n'a pas à vérifier la validité des différents intervalles de variations des variables (*no range checking*), la présence de cette commande permet d'accélérer le programme.

La saisie du nombre est protégée par la suppression du contrôle de validité des entrées/sorties (directive $I-). Si le format d'entrée est incorrect, le nombre est ignoré et garde sa valeur précédente. Par ailleurs si le nombre est supérieur ou égal à un million un message d'erreur est affiché.

Le type classique *INTEGER* n'acceptant que des nombres inférieurs à 37767, le programme définit un type *ENTIER* permettant de traiter des entiers écrits sur six chiffres (donc jusqu'à un million). Cette possibilité, que seul le Pascal UCSD permet, est extrêmement intéressante, bien que de tels entiers ne puissent pas être manipulés par tous les opérateurs arithmétiques. C'est le cas de *MOD* qui a du être simulé dans la procédure *ECRIRE*. ■

```
{$R-}
Program CONVERSION;
type entier=integer[6];

var nb:integer;
    nombre:entier;
    saisie:boolean;

  procedure ecrire(nbre:entier);
  var lquot:entier;
      quotient:integer;

    procedure moinsdemille(nb:integer);
    var quotient:integer;

      procedure moinsdevingt(nb:integer);

      begin {moinsdevingt}
      case nb of
        1:write('un');
        2:write('deux');
        3:write('trois');
        4:write('quatre');
        5:write('cinq');
        6:write('six');
        7:write('sept');
        8:write('huit');
        9:write('neuf');
       10:write('dix');
       11:write('onze');
       12:write('douze');
       13:write('treize');
       14:write('quatorze');
       15:write('quinze');
       16:write('seize');
       17:write('dix-sept');
       18:write('dix-huit');
       19:write('dix-neuf')
      end;
      write(' ')
      end;


    begin {moinsdemille}
    if nb >= 100 then
      begin
      quotient:=nb div 100;
      nb:=nb mod 100;
      if quotient > 1 then moinsdevingt(quotient);
      write('cent');
      if (quotient>1) and (nb=0) then write('s ')
                                 else write(' ')
      end;
    if nb >= 80
    then
      begin
      nb:=nb-80;
      write('quatre-vingt');
      if nb=0 then write('s ')
              else begin
                   write('-');
                   moinsdevingt(nb)
                   end
      end
    else
      if nb >= 60
      then
        begin
        nb:=nb-60;
        write('soixante');
        if(nb=1)or(nb=11) then write('-et-')
                          else if nb>0 then
                                          write('-');
        moinsdevingt(nb)
```

```
    end
  else if nb >= 20
    then
      begin
      quotient:=nb div 10;
      nb:=nb mod 10;
      case quotient of
        2:write('vingt');
        3:write('trente');
        4:write('quarante');
        5:write('cinquante')
      end;
      if nb=1 then write('-et-')
                else if nb>0 then write('-');
      moinsdevingt(nb);
      end
    else
      if nb > 0 then moinsdevingt(nb)
  end;


begin (ecrire)
if nbre>=1000 then
  begin
  lquot:=nbre div 1000;
  quotient:=trunc(lquot);
  nbre:=nbre-1000*lquot;
  if quotient > 1 then moinsdemille(quotient);
  write('mille ')
```

```
      end;
    nb:=trunc(nbre);
    moinsdemille(nb)
    end;


begin (programme principal)
page(output);
writeln('    Ecriture d''un entier positif');
writeln('        inferieur a un million');
repeat
  writeln;
  repeat
  saisie:=true;
    write('? ',chr(7));
    {$I-}
    readln(nombre);
    {$I+}
    if nombre>999999 then
    begin
      writeln('Nombre trop grand');
      saisie:=false
    end
  until saisie=true;
  ecrire(nombre);
  writeln
until nombre=0;
page(output)
end.
```

Copie d'écran sur imprimante Epson

Denise Amrouche

Copie sur imprimante d'un écran ne comportant que des caractères standards

L'espacement des lignes est mis à 8/72ème de pouce pour respecter au mieux la géométrie de l'écran.
Le début du programme n'est qu'un exemple, le sous-programme d'impression proprement dit commençant à la ligne 320.

```
100 REM *********************
105 REM * Impression d'ecran *
110 REM *      contenant     *
115 REM *   uniquement des   *
120 REM *     caracteres     *
125 REM *     standards      *
130 REM *********************
135 REM *      Basic TI      *
140 REM *********************
150 REM
160 CALL CLEAR
170 FOR I=1 TO 24
180 FOR J=1 TO 28
190 PRINT CHR$(64+I);
200 NEXT J
210 NEXT I
220 REM
230 REM SAISIE DE L'ORDRE D'IMPR
ESSION
240 REM ON DOIT APPUYER SUR LA T
OUCHE "P"
250 CALL KEY(3,K,S)
260 IF K<>80 THEN 250
270 GOSUB 320
280 STOP
290 REM
300 REM PROGRAMME D'IMPRESSION

310 REM  SI VOUS UTILISEZ UN BRA
NCHEMENT SERIE,METTEZ VOTRE ORDR
E HABITUEL SUIVI
 DE .CR
320 OPEN #1:"PIO.CR"
330 REM  ESPACEMENT DES LIGNES D
E 8/72 INCH POUR RESPECTER L'INT
ERLIGNE DE L'ECR
AN
340 REM AVANCE DE LIGNE ET RETOU
R CHARIOT
350 PRINT #1:CHR$(17)&CHR$(27)&"
A"&CHR$(8)&CHR$(10)&CHR$(13);
360 FOR I=1 TO 24
370 FOR J=1 TO 32
380 CALL GCHAR(I,J,C)
390 IF C>32 THEN 410
400 C=32
410 PRINT #1:CHR$(C);
420 NEXT J
430 PRINT #1:CHR$(13);CHR$(10);
440 NEXT I
450 PRINT #1:CHR$(27);"2";
460 CLOSE #1
470 RETURN
```

Copie sur imprimante d'un écran obtenu à partir d'un programme en Basic Etendu utilisant des "CALL CHAR"

Rappelons d'abord brièvement comment les caractères graphiques doivent être transmis à l'Epson.
Alors qu'en TI Basic on utilise un codage par tranches horizontales, l'Epson demande un codage par tranches verticales de 8 points. Ainsi, l'ensemble de 8 points représenté ci-dessous est associé, en base 2, au nombre 11010010 et en base 10 à $0 \star 1 + 1 \star 2 + 0 \star 2\hat{}2 + 0 \star 2\hat{}3 + 1 \star 2\hat{}4 +$



0⋆2^5+1⋆2^6+1⋆2^7 en allant de droite à gauche, soit 210; il nécessitera donc la transmission à l'imprimante de CHR$(210).

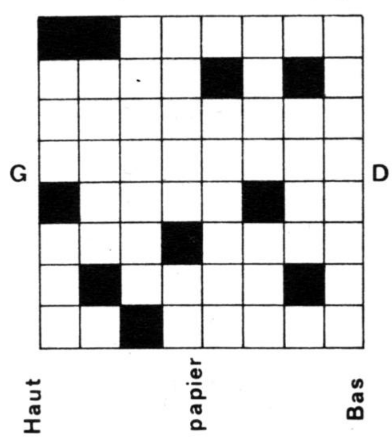
Le programme doit donc assurer la traduction. Cela est tout à fait possible mais, pour obtenir une copie d'écran agrandie (où chaque point de l'écran devient 4 points sur le papier), il me fallait patienter près d'une heure. D'où l'idée de raccourcir la phase de traduction en "tournant" l'écran : le haut de l'écran devenant le bord droit du papier. Il suffit alors de patienter environ un quart d'heure.

```
100 ! *********************
105 ! * Copie d'ecran sur  *
110 ! *  imprimante Epson  *
115 ! *********************
120 ! *    Basic Etendu    *
125 ! *********************
130 ! *  Si vous utilisez  *
135 ! *   une imprimante   *
140 ! *serie, reglez votre *
145 ! *   imprimante sur   *
150 ! *     huit bits      *
155 ! *********************
160 !
200 CALL CLEAR
210 CALL CHAR(58,"181818FFFF1818
1824242424242424240000FF0000FF00
0024242720203F")
220 FOR L=1 TO 24
230 FOR I=1 TO 4 :: CALL HCHAR(L
,7*I-4,57+I,5):: NEXT I
240 NEXT L
250 ! APPUYER SUR P POUR OBTENIR
 UNE COPIE
260 CALL KEY(0,K,S):: IF K<>80 T
HEN 260
270 GOSUB 310
280 GOTO 260
290 !
300 !   SOUS-PROGRAMME D'IMPRESS
ION
310 DATA 0,3,12,15,48,51,60,63,1
92,195,204,207,240,243,252,255
320 DIM A(15)
330 FOR I=0 TO 15 :: READ A(I)::
 NEXT I
340 ! POUR UN BRANCHEMENT SERIE,
METTEZ RS232.BA=9600.DA=8.CR
350 OPEN #1:"PIO.CR" :: PRINT #1
:CHR$(13);CHR$(27);"A";CHR$(8);
360 FOR CO=1 TO 32 :: G$,D$=""
370 FOR LI=24 TO 1 STEP -1 :: CA
LL GCHAR(LI,CO,CH):: IF CH=31 OR
 CH=32 THEN G$=G
$&RPT$(CHR$(0),16):: D$=D$&RPT$(
CHR$(0),16):: GOTO 420
380 CALL CHARPAT(CH,C$):: FOR L=
15 TO 16 :: FOR J=0 TO 7 :: W=AS
C(SEG$(C$,L-2*J,
1))
390 IF W>64 THEN W=W-55 ELSE W=W
-48
400 IF L=15 THEN G$=G$&RPT$(CHR$
(A(W)),2)ELSE D$=D$&RPT$(CHR$(A(
W)),2)
410 NEXT J :: NEXT L
420 IF LI=13 THEN GG$=G$ :: G$="
" :: DD$=D$ :: D$=""
430 NEXT LI
440 PRINT #1:CHR$(27);"K";CHR$(1
28);CHR$(1);GG$;G$;CHR$(13);CHR$
(10)
450 PRINT #1:CHR$(27);"K";CHR$(1
28);CHR$(1);DD$;D$;CHR$(13);CHR$
(10):: NEXT CO :
: CLOSE #1 :: RETURN
```

Comme pour le premier programme, le début n'est qu'un exemple, l'impression d'écran étant effectuée par le sous-programme final. Donnons-en le schéma :

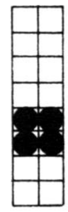
- Le travail se fait colonne par colonne, la première colonne d'écran devenant la première et la deuxième ligne sur le papier, puisque l'on a choisi d'agrandir le dessin.

- Partant du bas de la colonne, on lit successivement le code ASCII du caractère occupant cet emplacement (CALL GCHAR) puis le



patron de ce caractère (CALL PATTERN). Ce patron va nous donner 2⋆16 codes à envoyer à l'imprimante (2⋆2⋆8 à cause de l'agrandissement), 16 pour la première ligne mise dans G$ et 16 pour la seconde ligne mise dans D$ (partie droite).

Ce caractère étant sur l'écran, le premier code à transmettre à l'imprimante correspond à la dernière ligne, première moitié. Une fois agrandie (⋆2) et mise comme sur le papier, on doit obtenir :



On doit donc envoyer, pour un codage 2 sur le TI, les codes CHR$(12)&CHR$(12)&...
Ne vous affolez pas, le programme fait très bien tout cela pour vous.

Les chaînes en Basic TI ayant une longueur maximale de 255 caractères, chaque ligne sera envoyée à l'imprimante en deux parties :
GG$;G$ ou DD$;D$
et pour chacune on annonce à l'imprimante qu'on lui envoie une chaîne de 2⋆192, soit 384 caractères graphiques.

CHR$(27);"K";CHR$(128);CHR$(1)
car 384=128+1⋆256

Quelques précisions

- Les lutins ne seront pas reproduits sur l'imprimante.

- L'imprimante utilisée ne connaissant pas les couleurs, il n'y aura ni couleur ni nuance. Tout point "on", même s'il est en blanc sur fond noir, apparaîtra en noir sur fond blanc.
- Si votre dessin provient d'un programme en Basic simple (par exemple parce qu'il nécessite la redéfinition de nombreux caractères), vous pouvez adapter ce programme : si la mini-mémoire est insérée, vous disposerez encore de la fonction CALL CHARPAT; sinon, il faudra stocker dans un tableau les patrons des caractères au fur et à mesure de leur création.
- En tout état de cause, il faudra compter ici une bonne demi-heure pour obtenir votre copie d'écran. ■

Description du port d'entrée/sortie

Eliane Dacquay

En réponse à de nombreuses demandes de renseignements techniques, nous débutons ici une série d'articles sur l'architecture du TI-99/4A.

Ce premier "épisode" vous propose la description du port d'entrée/sortie, situé sur le coté droit de la console.

Dans les prochains numéros, nous examinerons le brochage du TMS 9900, son organisation, le câblage du port des modules "Solid State Software", etc...
Nous espérons ainsi satisfaire nos lecteurs qui, ayant du mal à trouver des périphériques pour leur ordinateur, souhaitent les réaliser eux-même. ■

Noms	Broches	Entrée/Sortie	Description
A0 (MSB)	31	Sortie	
A1	30	Sortie	Bus d'adresses
A2	20	Sortie	
A3	10	Sortie	
A4	7	Sortie	A0 à A15 représentent le bus d'adresses.
A5	5	Sortie	Ce bus fournit le vecteur d'adresse mémoire
A6	29	Sortie	sur 16 bits, au système mémoire externe,
A7	17	Sortie	lorsque MEMEN est actif. Le bit d'adresse
A8	14	Sortie	15 est aussi utilisé pour les sorties de
A9	18	Sortie	données CRU dans les instructions de
A10	6	Sortie	sorties du CRU.
A11	8	Sortie	
A12	11	Sortie	
A13	15	Sortie	
A14	16	Sortie	
A15/CRUOUT	19	Sortie	
D0	37	Entrée/Sortie	Bus de données
D1	40	Entrée/Sortie	
D2	39	Entrée/Sortie	D0 à D7 représentent le bus de données
D3	42	Entrée/Sortie	bidirectionel. Ce bus tranfère les données
D4	35	Entrée/Sortie	vers (en écriture) et depuis (en lecture)
D5	38	Entrée/Sortie	le système mémoire externe lorsque MEMEN
D6	36	Entrée/Sortie	est actif.
D7	34	Entrée/Sortie	
			Bus de Controle
MEMEN	32	Sortie	MEMory ENable. MEMEN indique un accès mémoire. Dérivée de la broche MEMEN du CPU. meme parité.
DBIN	9	Sortie	Data Bus IN. Quand actif (haut), les tampons de données et le 9900 sont en mode "entrée".
WE	26	Sortie	Write Enable. WE indique une écriture en mémoire.
MBE	28	Sortie	Memory Block Enable. MBE indique un accès dans le bloc mémoire >4000->5FFF (bas).
CRUCLK	22	Sortie	CRU CLocK. Inversion de la broche CRUCLK du CPU.
CRUIN	33	Entrée	CRU data IN. Ligne d'entrée des données vers l'ordinateur (CRU vers CPU).
			Controle mémoire
READY	12	Entrée	READY (quand MEMEN est actif) que la mémoire externe est prete pour un accès.
IAQ	41	Sortie	Instruction AcQuisition : indique que le CPU est en cours de saisie d'instruction, durant le cycle mémoire courant.
			Controle et temporisation
LOAD	13	Entrée	Quand actif, LOAD provoque l'exécution par le CPU d'une interruption non masquable. Les adresses mémoire >FFFC et >FFFE contiennent les vecteurs de la nouvelle mémoire de travail et du nouveau PC.
RESET	3	Sortie	Quand actif, RESET provoque la réinitialisation de l'ordinateur et des périphériques RESET doit etre actif pendant au moins cinq cycles d'horloge.
EXT INT	4	Entrée	EXTernal INTerrupt. Quand actif, EXT INT provoque l'exécution d'une interruption par le CPU.
03	24	Sortie	Phase 3 de l'horloge du CPU.
			Alimentation
GND	21,23 25,27		Référence de masse.
			Signaux du synthétiseur de parole
SBE	2	Sortie	Speech Block Enable. SBE indique un accès dans la mémoire du synthétiseur.
AUDIO IN	44	Entrée	Entrée audio depuis le synthétiseur.
+5	1		Tension d'alimentation (+5v nominale) pour le synthétiseur (50ma maximum)*.
-5	43		Tension d'alimentation (-5v nominale) pour le synthétiseur (50ma maximum)*.

* Une surcharge sur les broches 1 et 43 peut endommager la console.

Translet

Georges Dejolier

Suite à la rubrique "A vous de programmer" du numéro 4 de "99", je vous communique un programme écrit en **Basic Etendu**, que j'ai baptisé *Translet* (pour transposition en lettres).

Comme vous pourrez vous en rendre compte, l'utilisation en est très simple. L'appel de la routine constituant la partie active du programme requiert trois paramètres :

- le pays de travail (F pour la France et B pour la Belgique);
- le nombre à convertir (9 chiffres maximum);
- la chaîne de caractères destinée à recevoir le résultat. ■

```
100 ! ************************
110 ! *       TRANSLET       *
120 ! ************************
130 ! *     Basic Etendu     *
140 ! ************************
150 !
160 CALL CLEAR
170 INPUT "PAYS?":P$
180 INPUT "NOMBRE?":A
190 IF A>999999999 THEN 180
200 CALL TRANSLET(P$,A,Z$)
210 PRINT Z$
220 PRINT
230 GOTO 170
240 !
250 ! ***********************
260 ! *   Sous-programme    *
270 ! ***********************
280 !
290 REM PROGRAMME TRANSLET
300 REM AUTEUR GEORGES DEJOLIER
310 REM BRUXELLES
320 DATA " ET UN"," DEUX"," TROIS"," QUAT
RE"," CINQ"," SIX"," SEPT"," HUIT"," NE
UF"
330 DATA " DIX"," VINGT"," TRENTE"," QUAR
ANTE"," CINQUANTE"," SOIXANTE"," SEPTAN
TE"," QUATRE-VINGT"," NONANTE"
340 DATA " ONZE"," DOUZE"," TREIZE"," QUA
TORZE"," QUINZE"," SEIZE"," DIX-SEPT","
 DIX-HUIT"," DIX-NEUF"
350 SUB TRANSLET(R$,A1,Z$)
360 RESTORE 320
370 FOR I=1 TO 9
380 READ A$(I)
390 NEXT I
400 FOR I=1 TO 9
410 READ B$(I)
420 NEXT I
430 FOR I=1 TO 9
440 READ C$(I)
450 NEXT I
460 Z$=""
470 A(1)=INT(A1/1000000)
480 A(2)=INT((A1-A(1)*1000000)/1000)
490 A(3)=A1-A(1)*1000000-A(2)*1000
500 I=1
510 J=INT(A(I)/100)
520 K=INT((A(I)-J*100)/10)
530 L=A(I)-INT(A(I)/10)*10
540 IF A(I)=0 THEN 980
550 IF A(I)<10 THEN 880
560 IF A(I)<100 THEN 670
570 IF J=1 THEN 590
580 Z$=Z$&A$(J)
590 Z$=Z$&" CENT"
600 IF K>0 THEN 670
610 IF L>0 THEN 720
620 IF J=1 THEN 910
630 IF I=2 THEN 910
640 Z$=Z$&"S"
650 IF K=9 THEN 1020
660 GOTO 910
670 IF K=1 THEN 830
680 IF R$="B" THEN 710
690 IF K=7 THEN 1020
700 IF K=9 THEN 1020
710 Z$=Z$&B$(K)
720 IF L=0 THEN 790
730 IF L<>1 THEN 750
740 IF K=8 THEN 770
750 Z$=Z$&A$(L)
760 GOTO 910
770 Z$=Z$&" UN"
780 GOTO 910
790 IF K<>8 THEN 910
800 IF I=2 THEN 910
810 Z$=Z$&"S"
820 GOTO 910
830 IF L<>0 THEN 860
840 Z$=Z$&B$(1)
850 GOTO 910
860 Z$=Z$&C$(L)
870 GOTO 910
880 IF L>1 THEN 720
890 IF I=2 THEN 910
900 Z$=Z$&" UN"
910 IF I=1 THEN 950
920 IF I<>2 THEN 1010
930 Z$=Z$&" MILLE"
940 GOTO 980
950 Z$=Z$&" MILLION"
```

```
960 IF A(I)<2 THEN 980
970 Z$=Z$&"S"
980 IF I=3 THEN 1010
990 I=I+1
1000 GOTO 510
1010 SUBEXIT
1020 Z$=Z$&B$(K-1)
1030 IF L<>0 THEN 1060
1040 Z$=Z$&B$(1)
1050 GOTO 910
1060 IF L>1 THEN 1090
1070 IF K=9 THEN 1090
1080 Z$=Z$&" ET"
1090 Z$=Z$&C$(L)
1100 GOTO 910
1110 SUBEND
```

Un mode test sur "Chisholm Trail"

Antoine Heudes

Les modules "Munch-Man" et "Alpiner" ne sont pas les seuls à disposer d'un mode test (voir numéros 3 et 4 de "99").

Après le dessin de la "tête de veau" de Texas Instruments, il faut appuyer successivement sur * # * (SHIFT-8, SHIFT-3, SHIFT-8) pour obtenir ceci :

- Screen num ones digit ? : introduire le chiffre des unités pour le Nème tableau
- Screen num tens digit ? : introduire le chiffre des dizaines pour le Nème tableau
- Ship count ? : introduire le nombre de séries de vaisseaux, sachant qu'une série est égale à quatre vaisseaux.

"DISPLAY" et "ACCEPT" avec le module "Statistic"

Omar Belmokar

Les fonctions supplémentaires, disponibles à partir du Basic TI lorsque le module "PRK" est enfiché, sont aussi résidentes dans le module "Statistic".

Ces fonctions (CALL A, CALL D) sont expliquées en détail dans le précédent numéro de "99 Magazine".

Encore une bonne adresse

Qui n'a jamais poussé de grands cris quand, voulant taper un "+", il a fait FCTN +, et s'est retrouvé devant la mire, le programme s'étant envolé !

Le contenu de l'adresse >83C2 (−31806) détermine l'activité ou la non-activité de la touche "QUIT" (FCTN +). Si on place la valeur >10 (16 en décimal) dans cette adresse, on inhibe la touche "QUIT", sans effets secondaires semble-t-il.

Pour ce faire, en Basic TI, vous insérez la Mini-mémoire et vous tapez :

CALL LOAD (−31806,16)

En Basic Etendu, *uniquement si vous disposez d'une extension mémoire*, vous tapez :

CALL INIT :: CALL LOAD (−31806,16)

Plus de mémoire

Lorsque l'on débute en Assembleur avec la Mini-mémoire, les quatre Ko de mémoire vive sont généralement suffisants mais, dès que l'on commence à maîtriser ce langage, nous avons envie de réaliser des programmes plus ambitieux.

Un programme en Assembleur, utilisant le mode graphique ou texte pour l'affichage, laisse une grande partie de la mémoire vidéo inutilisée (plus de 10 Ko). Malheureusement, pour qu'un programme puisse être exécuté, il faut qu'il soit dans une zone de mémoire vive directement adressable par le microprocesseur.

Cependant, on peut très bien utiliser la mémoire vive du processeur vidéo pour le stockage temporaire de certaines routines. Le programme pourrait être utilisé selon la technique de "l'overlay dynamique", dans laquelle les routines se trouvent en mémoire vive uniquement au moment de leur exécution. Cette solution permet l'écriture de programmes pouvant occuper jusqu'à 14 Ko environ, seul le segment principal étant en permanence dans la RAM du TMS 9900.

Il est évident que cette technique pose de nombreux petits problèmes : gestion des routines, sauvegarde et chargement des programmes, appel et exécution des routines à partir du programme principal, etc... mais le fait de pouvoir "récupérer" une dizaine de Ko vaut bien un petit effort.

Un article sera consacré à cette technique, qui peut aussi intéresser ceux d'entre vous qui disposent d'une extension 32 Ko (on a jamais trop de mémoire), dans un prochain numéro.

Le Club Ordinobis

Nous sommes un club (association loi 1901) à vocation *toutes informatiques*, avec, vu le prix et les capacités du TI-99/4A, une dominante actuellement pour ce matériel.

Notre club a été fondé en Juin 83 et est organisé pour accueillir toute personne qui, dans la région, s'intéresse à l'informatique.

Nos adhérents, avec leur carte du club, reçoivent un bulletin bimestriel, et ont directement accès à notre programmathèque (plus de 150 logiciels, surtout pour TI).
Grâce à des accords avec des fournisseurs, les membres du club peuvent obtenir des réductions sur présentation de leur carte, cela pour 120 F par an.

Actuellement, en plus de nos permanences téléphoniques du Lundi au Mercredi, nous offrons deux jours de réunions : réunion informelle (pour faire fuser les idées) le Jeudi soir, et réunion formelle (cours de Basic pour débutants) le Samedi après 14 heures.

Notre adresse :

Club Ordinobis
Boîte postale 57
33603 Pessac

Téléphone :

Lundi après 17h - (56) 30.55.01
Mercredi après 17h - (56) 45.55.16 ■

Courrier des lecteurs

Alexandre Duback

Par quel miracle "99 Magazine" survit-il ? J'espère que tous les "Texans" vous resteront fidèles car votre magazine devient notre Bible... Avant de vous mettre à contribution, je vous livre l'information suivante concernant le logiciel "Gestion Privée".

La déclaration du nom de l'imprimante en sortie parallèle, pour obtenir une copie d'écran, doit avoir la syntaxe "PIO." au lieu de "PIO". En effet, si vous ne faites pas suivre la déclaration PIO d'un point, la communication avec l'imprimante n'existe pas.

Passons à ma question qui concerne encore l'utilisation d'une imprimante, la GP 100A de Seikoska en la circonstance. Si les déclarations en mode "normal" ou "dilaté" ne me posent pas de problèmes, je suis incapable d'utiliser le mode graphique. Il semble que la commande CHR$(8) ne soit pas suffisante pour obtenir les caractères redéfinis par l'instruction "CALL CHAR". Pouvez vous m'indiquer la marche à suivre pour que l'imprimante accepte ces caractères. Je dois avouer que la solution à ce problème me permettra sans doute de récupérer quelques heures de sommeil.

Daniel Leligny - 190 Boulevard Jacques Cartier - 35100 Rennes

Il n'y a pas de miracle ! Seule notre détermination à continuer et la confiance de nos lecteurs font que vous pouvez toujours lire "99 Magazine".

Nous avions déjà rencontré ce problème d'imprimante avec l'Editeur/Assembleur, et nous en avons trouvé la cause : la routine utilisée pour le pilotage d'un périphérique (DSRLNK) cherche le ou les points séparant le nom de l'appareil et le ou les éventuels paramètres (PIO.LF ou RS232.BA=9600.PA=E, par exemple). Si cette routine ne trouve pas de point, elle ne reconnaît pas le périphérique.

Comme vous le supposez, il ne suffit pas de faire passer l'imprimante en mode graphique pour transmettre des caractères redéfinis avec CALL CHAR. La copie d'écran sur une GP 100 fait partie de nos projets, mais vous pouvez déjà vous inspirer du programme de Denise Amrouche, publié dans ce numéro, et concernant l'imprimante Epson.

Amateur de vidéo, j'aimerais vous poser une question. Est-il possible de faire un générique à partir du TI-99/4A (c'est-à-dire une image ou un texte que je puisse enregistrer sur un magnétoscope). J'ai entendu dire qu'avec certains micro-ordinateurs, on pouvait faire de l'affichage sur un téléviseur par surimpression.

Si vous pouviez m'aider à élucider cette question, je vous en serais reconnaissant.

Yves-Marie Lamirault - 8 résidence des Hauts de Gien - 45500 Gien

Il est effectivement possible d'enregistrer des images issues du TI sur un magnétoscope. Le schéma, situé au bas de la page 64 du numéro 4 de "99", était d'ailleurs destiné à expliquer la marche à suivre pour arriver à ce résultat. Malheureusement, un violent coup de ciseaux au moment du maquettage a fait disparaître le texte accompagnant le dessin.

Pour enregistrer les images du TI, il faut se munir d'un modulateur UHF, que l'on raccorde sur la prise d'antenne du magnétoscope. Vous pouvez utiliser le modulateur proposé par Texas Instruments ou, si vous ne le trouvez pas, un convertisseur Péritel/UHF de marque "VIDEO MATCH" distribué, entre autres, par Pentasonic - 34 rue de Turin - 75008 Paris. Ce convertisseur coûte environ 440 F.

Nous ne pensons pas qu'il soit possible de superposer des images issues du TI à des images télévisées, à moins qu'un de nos lecteurs connaisse le truc. A suivre...

Je voudrais savoir si votre société dispose encore actuellement des programmes de Basic Etendu pour TI-99/4A, ainsi que le jeu Foot Ball.

Jean-Philippe Ravelet - 25 rue Alfred de Vigny - 54630 Richardmeuil

Nos activités se limitent à la publication de "99 Magazine" et de "Pom's" (pour les utilisateurs d'ordinateurs Apple). Les seuls logiciels que nous proposons pour le TI sont ceux qui accompagnent la revue.

J'achète "99 Magazine" et je le trouve très bien, mais j'ai quand même quelques griefs et doléances à formuler :

1) Les préfixes. Pour les petites annonces, les gens mettent un numéro de téléphone, sans spécifier le déplacement et même le pays. "99" pourrait insister sur ce fait.

2) Le coût. Le magazine est cher, par rapport à d'autres, ne pourrait-on y inclure plus de publicité, cela ferait diminuer son prix.

3) La touche RESET. Ne pourrait-on nous l'expliquer, quand on le demande à des vendeurs, ils ne savent pas, ou bien il faut payer (sans doute).

4) Un truc que j'aimerais voir développer : sur certains ordinateurs, en enfonçant deux touches simultanément, on fait apparaître un mot entier. Peut-on, avec la mini-mémoire, imiter cela.

5) Certains ordinateurs emploient des "super expenders", sortes de dominos permettant de brancher plusieurs jeux, le Basic Etendu, la mini-mémoire, etc... sans devoir retirer l'un pour l'autre, j'ai posé la question et on m'a dit non. Est ce uniquement parce que cela n'est pas concevable.

Monsieur Simon

Voilà un bon nombre de questions et suggestions, auxquelles nous allons essayer de répondre le plus justement possible.

1) Effectivement, le manque d'indicatif peut décourager les éventuels acheteurs (ou vendeurs), mais nous ne pouvons reproduire dans la revue plus que l'on nous communique.

2) "99 Magazine" est uniquement destiné aux utilisateurs de matériel Texas Instruments. Pour cela, la revue est considérée par l'administration française comme un organe "maison", alors que nous ne sommes absolument pas liés à cette société (vous ne seriez pas en train de nous lire). Nous ne bénéficions donc pas des tarifs consentis à la presse écrite, en particulier pour ce qui concerne les tarifs postaux.

D'autre part, nous ne pouvons pas obliger d'éventuels annonceurs à faire de la publicité dans "99". Par contre, si vous avez quelque chose à vendre, vous pouvez contacter notre régie publicitaire, et soyez sûr que votre demande sera prise en considération.

Enfin, "99" est peut-être chère par rapport à d'autres revues, mais si vous comparez les nombres de pages utiles aux possesseurs d'un TI...

3) Donner une explication précise du fonctionnement de la touche "RESET" proposée dans le numéro

3, et améliorée dans le numéro 4, occuperait de nombreuses pages. De plus, un tel article serait absolument incompréhensible pour les lecteurs ne possédant pas une formation d'électronicien.

4) Comme sur tous les ordinateurs, les mots clés du Basic sont "tokenisés". C'est-à-dire qu'ils sont codés sur un caractère ASCII, généralement supérieur à 127 (caractères de contrôle). Tapez la ligne suivante :

10 REM

puis enfoncez la touche CTRL-H, et appuyez sur "ENTER".
Si vous éditez cette ligne, vous verrez apparaître :

10 REM RETURN

On peut en déduire que le code interne de "RETURN" est 136 (CTRL-H est équivalent à 136)

Ceci ne présente pas un grand intéret puisqu'il est plus facile de taper un mot clé que de rechercher le code équivalent dans une table. De plus, ces codes ne sont acceptés qu'à la suite de "REM". La saisie d'un programme par cette méthode serait certainement plus longue que par la méthode classique (recherche des codes, supression des REMs, erreurs possibles...).

5) Un connecteur multiple existe aux Etats Unis. Cet appareil accepte jusqu'à trois modules qui sont sélectionnés par un commutateur. Le fait de changer manuellement de module ne représente pas un travail considérable et nous considérons que ce connecteur est à ranger au nombre des gadjets.

N'ayant plus à faire l'éloge de votre magazine, je désirerais solliciter vos services pour un renseignement.
Je suis néophyte en informatique et je me suis dirigé vers le TI-99. Grâce à votre magazine, j'ai pu apprendre la plupart de mes connaissances dans ce domaine.

Dans votre numéro 2, l'article consacré au TI-Writer a attiré mon attention au sujet des 40 caractères disponibles en Basic.

Pourriez-vous me faire savoir comment acquérir cet ensemble de programmes ?

Roland Lardin - 113 rue de la Vallée - 6290 Malinnes - Belgique

Mettons tout de suite les choses au point : le Basic est allergique au mode 40 colonnes. S'il est possible de passer en 40 colonnes, cela ne sert à rien puisque l'affichage se fait en incolore sur cyan, ce qui n'est pas la meilleure solution pour la lisibilité. D'autre part, s'il est possible d'appeler une routine en assembleur pour changer les couleurs, et ensuite revenir au Basic, l'affichage devient incohérent puisque l'interpréteur conserve ses habitudes (32 colonnes avec HCHAR ou VCHAR et 28 colonnes avec PRINT).

Il vaut mieux réserver ce type d'affichage aux programmes en assembleur. Vous avez des exemples concrets du mode 40 colonnes avec le Mastermind du numéro 3 et le désassembleur publié dans ce numéro.

Est-il possible de relier le TI-99 à une imprimante Star Gemini 10, sans passer par le boîtier d'extension, cette imprimante possédant, en option, une interface RS232.

Roger Feyssaguet - Chizé - 79170 Brioux sur Boutonne

Voilà bien la preuve que les notices et publicités des constructeurs ne sont pas toujours très claires. En effet, cette imprimante est interfacée pour recevoir des signaux provenant d'une carte RS232, mais n'est pas équipée d'une telle interface.

Permettez-moi, tout d'abord, de vous féliciter pour la qualité de votre revue qui permet aux possesseurs de TI-99 de découvrir les secrets de la programmation.

Cependant, depuis l'arrêt de la fabrication de ce micro, il est devenu difficile de se procurer le module "Basic Etendu", et il serait intéressant de savoir si certaines de ses instructions ont un équivalent en Basic TI.

Par exemple, les instructions "AND" et "OR", dans les lignes suivantes :

300 IF N=Z AND T>N1 THEN 302
ou
260 IF Z<1 OR Z>10 THEN 240

Christian Pino - 23 Avenue Jean Jaures - 34370 Cazouls les Beziers

Les "ANDs" et "ORs", losqu'ils sont utilisés de cette façon, peuvent être remplacés par "⋆" (pour AND) et "+" (pour OR). Ainsi, les lignes citées dans votre courrier deviennent :

300 IF (N=Z) ⋆ (T>N1) THEN 302
et
260 IF (Z<1) + (Z>10) THEN 240

J'ai pu constater qu'avec le Basic Etendu, trois routines ne sont plus disponibles par rapport au module Editeur/Assembleur. Il s'agit des routines suivantes : GPLLNK, DSRLNK et LOADER.

Peut-on écrire en assembleur ces trois routines pour une utilisation à partir du Basic Etendu ?

Bernard Bezuel - 7 rue du Poirier Fourrier - 95100 Argenteuil

En assembleur, tout est possible (dans les limites des possibilités de la machine).

Le meilleur moyen, pour récupérer ces routines est d'utiliser le désassembleur proposé dans ce numéro. Après les quelques modifications indispensables (mémoire de travail de l'interpréteur GPL, par exemple), vous pourrez les réassembler et les utiliser à partir du Basic Etendu. Toutes les routines système, avec l'Editeur/Assembleur, se situent entre >2000 et >26FF.

Après de vaines recherches, j'ai fini par vous écrire pour un problème qui me tracasse depuis quelque temps.
Existe-t-il un moyen pour inhiber la touche "CLEAR" qui arrête un programme en cours.

Frédéric Rejol - Le Tivoli - 42820 Ambierle.

S'il est possible d'inhiber la touche "QUIT" (voir rubrique trucs et astuces), nous n'avons pas de solution, en Basic TI, pour la touche "CLEAR".
Par contre, en Basic Etendu, il suffit d'utiliser l'instruction "ON BREAK NEXT".

Je voudrais écrire, avec des lutins, le mot "LETTRES", mais cela "coince" après avoir affiché trois ou quatre lettres. Pouvez-vous me donner la solution.

Jean-Claude Dadone - 23 Avenue de la Salis - 06600 Antibes

Lorsque plus de 4 lutins se trouvent sur la même ligne d'écran, ceux qui ont le poids le plus fort (numéro) n'apparaissent pas. Il n'y a pas vraiment de solution; pour que le mot "LETTRES" soit affiché entièrement, il faudrait opter pour une disposition verticale ou en escalier.

Vous pouvez consulter l'article de Julien Thomas sur les lutins, publié dans le numéro précédent, pour obtenir de plus amples informations.

L'initiation à l'informatique laisse trop de coté le "Hardware". Si l'on considère qu'il suffit de connaître la façon de programmer, en quelque langage que ce soit, on peut en conclure (avec un peu d'exagération) qu'un téléspectateur qui sait commander son téléviseur connait "la télévision".

A. Maillet - Lieu dit "Guesdin" - 17610 Chaniers

Nous ne sommes pas vraiment d'accord avec vous, et ne pensons pas qu'il soit indispensable de connaître

Suite page 67

Suite de la page 65
la structure d'un ordinateur pour l'utiliser ou le programmer.

Nous ne citerons que le cas de Von Neumann, qui fut à la base de tous les algorithmes utilisés sur les ordinateurs classiques (comme le TI), sans être pour autant spécialiste de l'électronique. ■

Petites annonces (gratuites)

Recherche programmes sur K7. Echange prolongé. Hervé Kratiroff - 60, rue Mazgnoo - 69003 Lyon - 16 (70) 871.02.95.

Vends cassettes : La tombe du sorcier, Tractors folies (Basic Etendu), Golf et Basic Etendu par soi-même. David Germain-Dobin - 137 Avenue de Wagram - 75017 Paris - 766.37.81.

Recherche logiciels, modules et programmes sur cassettes. Olivier Serreault - Chemin du plan de Leydet - 04200 Sisteron.

Recherche interface RS232 et Terminal emulator 2. Echange trucs et programmes assembleur, Basic ou Basic Etendu. Philippe Pair - 6 Avenue St Sébastien - 54600 Villers-les-Nancy - (8) 327.74.73.

Vends 200 F cassette MASH neuve. 16 (58) 43.64.30.

Recherche module "Terminal Emulateur II". Omar Belmokar - 6 Allée S. de Brosse - 93150 Le Blanc Mesnil.

Achète carte contrôleur, carte RS232, lecteur de disquette intégré. Yves Jacquet - 124 rue de la Convention - 75015 Paris - 558.10.07.

Achète pépiphérique PHP 1200. P. Loraux - 2 rue Legraverend - 75012 Paris - (1) 345.38.84.

Achète boîtier d'ext., carte 32K, carte RS232, contrôleur et lecteur de disquettes, de toute urgence. Christophe Auzzino - Maison neuve Guillevin - Avenue de Bretagne - 56510 St Pierre-Quiberon.

Recherche module "Adventure" ainsi que synthétiseur de paroles - Marc Mesguini - 358.34.04.

Recherche Editeur/Assembleur avec documentation - Menasse - (24) 40.12.19 à 08500 Revin.

Recherche tous périphériques d'occasion, Basic Etendu, Mini-mémoire, modules de jeux - Daniel Bauwens - 92, rue Delsamme - B-7060 Stepy Bracquenies - Belgique.

Vends 2 modules éducation ayant très peu servi, au prix de 110 F chaque : Dragon savant et Addition canon - (6) 010.07.72 après 18h.

Achète module "Basic Etendu". Recherche programmes mathématiques et biologie. Jean-Yves Nicolas - 24 rue Troudet - 56310 Bubry.

Achète module "Basic Edendu". Joseph Caubisens - Résidence Lautrec - 10 rue Casteret - 64000 Pau.

Achète bon prix : boîtier d'extension, carte 32 K et carte RS232. Jean Gaillard - 43 Allée Paul Gauguin - Cidex 8 - 94450 Limeil-Brevannes - 569.01.76 après 20h.

Recherche boîtier d'extension avec si possible la carte 32 K, pour une somme abordable. Christophe Roux - 4 Impasse des Cigols - 33320 Eysines.

Recherche module "Basic Etendu" - Carte RS232 et éventuellement imprimante. Martin - (71) 50.39.29.

Recherche module "Basic Etendu". Patrick Bertin - 16 Place du Bois Perchix - 95220 Herblay.

Achète Basic Etendu + manuel (si possible en français). Philippe - 73 route de Sierck - Koenigsmacker - 57110 Yutz - (8) 250.01.46.

Recherche pour recopier ou acheter, manuel Basic Etendu en français. En échange, donne plein de programmes. Jean-Jacques Denis - 16 (85) 53.13.37.

Vends TI-99/4A, boîtier d'extension, contrôleur, lecteur de disquettes, carte 32 Ko, Assembleur, manettes de jeux, 4 modules de jeux, disquettes + livres + programmes. Le tout : 9000 F. 334.09.80 après 22h.

Achète module "Terminal Emulator II". Bruno Momerency - 33 Avenue Paul Bert - 92190 Meudon - (1) 534.85.13 après 18h.

Echange Mini-mémoire contre Terminal Emulator II. Jean-Pierre Diot - 47 rue Jacquart - 51100 Reims.

URGENT : Recherche cartouches, cassettes, jeux pour TI, ainsi que périphériques. Thierry Ducatez - Résidence Provence - 2/51 rue Louis Bergot - 59155 Fâches Thumesnil - (20) 53.83.29. ■

Le Club Ticazur

Ticazur, club d'une cinquantaine de membres, régi par la loi de 1901, existe depuis Juin 1983 et est consacré au TI-99/4A.

Jean Marin a assuré, durant la première année, un cours d'initiation au Basic selon une méthode originale d'enseignement de dix cassettes assurant 20 heures de cours.

Ce cours est repris depuis début mai par Paul Pacotto, professeur d'électronique à l'IUT de nice, de 16 à 17H00 tous les samedis au Cercle Démocratique de la Colle-sur-Loup.

De 17 à 18H00, des échanges sont organisés ainsi que des démonstrations d'autres langages ou ordinateurs.

De 18 à 19H00 se déroule, depuis le 12 mai 1984, le nouveau cours de Jean Marin sur l'utilisation de la Mini-mémoire dont, grâce à ses longues heures de travail, aucune des possibilités, que ce soit à partir du Basic, avec *EASY BUG* ou en Assembleur, ne seront négligées.

Sur le plan départemental, *Ticazur* est sans conteste le club le plus actif et, grâce à notre animateur principal, les limites du TI-99/4A reculent chaque jour davantage.

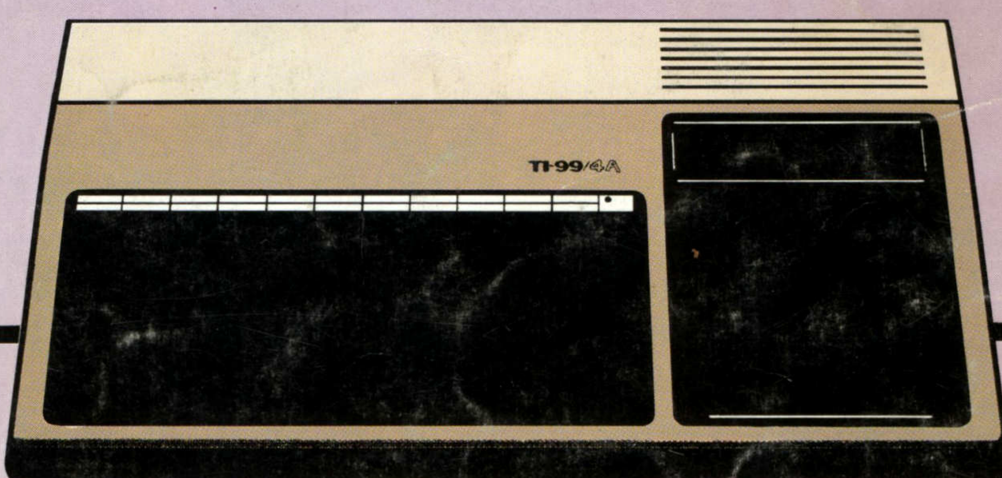
Gérard Baroni
Président de Ticazur

Siège social :
Villa Héléna
Les Campons
06480 La Colle sur Loup
(93) 22.63.16 ■

CASSETTE
D'ACCOMPAGNEMENT
DISPONIBLE

99 MAGAZINE

LE BASIC A LA LOUPE
A L'HEURE DU PASCAL
MINI-MEMOIRE MOINS "MINI"
ASSEMBLEUR...



NUMÉRO 6 - Trimestriel - Septembre 1984 - 40 F

ISSN 0759-9005

99 MAGAZINE

Sommaire

	Page
Editorial par Hervé Thiriez	5
Des chiffres et des lettres par Edouard Krok	6
A l'heure du Pascal par Gérard Santraille	13
Kong par François Blanchard	14
Mini-Mémoire moins "mini" par Guy Fix	17
Edipage par Logik	19
Caractères ASCII dans la page graphique par Jean-Luc Bazanegue	29
Musik par Michel Legault	27
Copie d'écran sur Seikosha GP100 par Mansur Tufail	30
Nouvelles fonctions avec la Mini-mémoire par Denise Amrouche	31
Le Basic à la loupe par Marianne Sutz	35
Star Trek par Jean-Philippe Guillemant	39
Ecriture directe dans les registres VDP par Gérard Baroni et Jean Marin	51
My tailor was rich par T. Gallier	53
Le code de la route par Jacques Keck	56
Un double face sur votre TI par Sophie Ehster	60
Courrier des lecteurs par Alexandre Duback	63
Petites annonces	64

Les annonceurs
L'Ordinateur Individuel : page 2 - SICOB : page 65 - La Règle à calcul : page 67 - PSI Diffusion : page 68.

Editions MEV - 64-70, rue des Chantiers - 78000 Versailles

(Dépôt légal : 3e trimestre 1984)

Editorial

Cela fait maintenant presqu'une année que Texas Instruments a interrompu la fabrication du TI-99/4A. Cependant, nous sommes toujours là car nous vous avions fait la promesse de ne pas vous abandonner, mais aussi parce que notre ordinateur est loin d'être mort. La Règle à calcul, célèbre revendeur parisien, nous le prouve en sortant une extension de mémoire, une interface parallèle et divers logiciels parmi lesquels nous trouvons la Mini-mémoire et le Basic-Etendu.

La cassette d'accompagnement de ce numéro, contrairement à nos habitudes, comporte des programmes différents sur les deux faces : cela est dû au fait que certains programmes proposés dans ce numéro sont très longs, alors que des problèmes techniques liés à la duplication de cassettes "informatiques" nous interdisent de dépasser une certaine longueur de bande magnétique. Je ne pense pas que cela soit un problème car le taux de défection est vraiment très faible.

Nous essayons de mettre dans chaque numéro des programmes et articles de tous niveaux et de toutes natures. Cependant, si vous désirez voir paraître un article sur tel ou tel sujet, ne manquez pas de nous le faire savoir : nous sommes à votre disposition.

*Passons maintenant au contenu de ce numéro. **Edouark Krok** nous propose une fois de plus un excellent programme de son cru, qu'il a intitulé "Des chiffres et des lettres", pendant que **François Blanchard** vous entraîne dans un combat sans merci contre l'abominable Kong.*

***Gérard Santraille,** toujours fidèle au poste, n'a pas délaissé son cher Pascal puisqu'il vous propose deux programmes susceptibles de vous donner l'heure. Pour les mordus de l'Assembleur, - langage primaire mais ô combien efficace, **Jean-Luc Bazanegue** vous communique une routine propre à afficher des caractères ASCII dans la page graphique.*

*Les bricoleurs pourront s'en donner à cœur joie avec l'article de **Guy Fix,** qui vous propose d'étendre la mémoire de votre Mini-mémoire, mais aussi avec celui de **Sophie Ehster** qui vous démontre qu'il est parfaitement possible de raccorder des lecteurs double face sur votre TI, à conditions, toutefois, de disposer du contrôleur de disquettes Texas Instruments.*

***Marianne Sutz** nous a concocté un très intéressant article sur l'organisation en mémoire du Basic TI, alors que **Gérard Baroni** et **Jean Maria** - les deux animateurs du club Ticazur - se sont penché sur la Mini-mémoire et ont découvert un moyen de détourner l'instruction CALL PEEK de son rôle habituel.*

***Logik** (il s'agit certainement d'un pseudonyme...) met à votre disposition un programme de traitement de texte écrit en Basic TI, pendant que **Michel Legault,** dans un tout autre registre, utilise les possibilités musicales du TI pour le transformer en un outil de création musicale. **Jack Keck,** quant à lui, vous offre un programme qui vous permettra de réviser le code de la route.*

*Les possesseurs d'imprimante Seikosha peuvent désormais s'adonner aux joies de la copie d'écran, grâce au programme de **Mansur Tufail. Denise Amrouche** vous propose quelques routines en Assembleur afin d'améliorer l'affichage du Basic TI, alors que **T. Gallier,** en cette période de rentrée scolaire, vous invite à participer à un cours d'anglais.*

*Nous ne conclurons pas sans citer le conséquent mais excellent jeu d'aventure de **Jean-Philippe Guillemant,** et la rubrique dédiée à votre courrier, toujours assurée par **Alexandre Duback.***

Hervé Thiriez

Rédaction en chef-Directeur de la publication : Hervé Thiriez. **Comité de rédaction :** Jean-Luc Bazanegue, Gérard Santraille. **Dessins :** Laurent Bidot. **Ont collaboré à ce numéro :** Denise Amrouche, Gérard Baroni, François Blanchard, Jean-Luc Bazanegue, Alexandre Duback, Sophie Ehster, Guy Fix, T. Gallier, Jean-Philippe Guillemant, Jacques Keck, Edouard Krok, Michel Legault, Logik, Jean-Marin, Gérard Santraille, Marianne Sutz, Mansur Tufail.

Editions MEV - 64-70, rue des Chantiers - 78000 Versailles.

Régie publicitaire : Force 7 - Anne Jourdan - 5, place du Colonel Fabien, 75010 Paris - Tél. : 240.22.01.

Diffusion auprès des boutiques et librairies : PSI Diffusion - BP 86 - 77402 Lagny-sur-Marne Cedex.

Composition : Télécompo - 13-15, avenue du Petit Parc - 94300 Vincennes.

Impression : Imprimerie Rosay - 47, avenue de Paris - 94300 Vincennes - Tél. : 328.18.63.

Des chiffres et des lettres

Edouard Krok

Il me semble superflu de présenter ce jeu qui passionne, chaque soir, des milliers de téléspectateurs et qui a fait naître de nombreux clubs dans toute la France.

Le programme que je vous propose accepte jusqu'à quatre joueurs et vous permet de choisir entre le mot le plus long, le compte est bon et le jeu des chiffres et des lettres. Vous décidez aussi de la durée du jeu.

Ce programme respecte les mêmes pourcentages de tirage des lettres et des chiffres que le jeu télévisé; il serait en effet aberrant de tirer autant de "W" que de "R" ou de "S", de même pour les chiffres. L'ordre des joueurs, qui revêt une certaine importance si l'on joue à plus de deux, est déterminé par tirage au sort en début de partie. Le nombre de tours, ainsi que l'alternance des jeux de chiffres et de lettres sont combinés pour laisser des chances identiques à chaque joueur, quel que soit le nombre de participants.

J'ai prévu la possibilité d'interrompre le chronomètre en appuyant sur n'importe quelle touche; il me paraît en effet assez agaçant, lorsque l'on joue seul, de devoir attendre quarante secondes si on trouve le bon compte en cinq secondes (oui, cela arrive quelquefois !).

La question est plus délicate si on est plusieurs : on peut convenir, en début de partie, d'attendre systématiquement le temps réglementaire de 45 secondes, ou de n'interrompre le chronomètre qu'avec l'accord des participants.

Dans tous les cas, votre temps de réponse sera limité et vous serez sanctionné si vous le dépassez. Si ce délai de réponse vous paraît trop court (ou trop long), il vous suffira d'augmenter (ou de diminuer) la valeur de la variable *DL*, à la ligne 100.

Le jeu du mot le plus long ne me semble pas poser de problèmes particuliers, étant entendu que vous choisissez une voyelle ou une consonne en appuyant sur *V* ou *C*; mais si vous mettez trop de temps pour vous décider, c'est le programme qui tranchera. En revanche, il vous laissera tout le temps qui vous sera nécessaire si vous désirez contrôler la validité d'un mot dans le dictionnaire. Notez aussi qu'il vous faudra taper sur une touche quelconque du clavier pour poursuivre la partie.

Pour le jeu du compte est bon, quelques explications vous feront mieux saisir la souplesse de ce programme. Les opérateurs utilisables pour donner votre solution sont :

- addition : +
- soustraction : −
- multiplication : ⋆ ou X
- division : / ou :
- résultat : =

N'oubliez pas, bien sûr, d'appuyer ‡r la touche *SHIFT* lorsque vous utilisez +, −, ⋆ ou :.

Exemple :

nombre à trouver - 635
Chiffres tirés - 6 - 10 - 8 - 3 - 100 - 2

Première solution valide :
100 + 8 = 108
108 − 3 = 105
105 ⋆ 6 = 630
10 : 2 = 5
630 + 5 = 635

Seconde solution valide :
100 + 8 − 3 ⋆ 6 =630
10 : 2 + 630 = 635

Vous remarquerez, dans le deuxième exemple, que les opérations sont évaluées dans l'ordre où elles sont posées, et que les parenthèses ne sont pas acceptées.

Si aucun joueur ne trouve le bon compte, le programme vous proposera une solution parmi d'autres possibles.

Si vous désirez jouer exactement comme au jeu télévisé, il vous suffira de choisir le jeu des chiffres et des lettres à deux joueurs et en vingt tours.

■

```
10 REM ********************
20 REM * JEU DES CHIFFRES *
30 REM *  ET DES LETTRES  *
40 REM ********************
50 REM *     BASIC TI     *
60 REM ********************
70 REM *                  *
80 REM *   EDOUARD KROK   *
90 REM *                  *
91 REM *                  *
92 REM ********************

100 DL=200
110 CALL SCREEN(8)
120 RANDOMIZE
130 OPTION BASE 1
140 DIM SC(4),A(4),S$(5),LG(4)
150 DEF V=INT(J/2.1)*5+2
160 DEF H=(J-INT(J/2.1)*2)*14-9
162 DEF Z=INT(6*RND)+1
164 DEF N(X)=INT(-10*RND)*(X>1)+INT(-4*RN
D)*25*(X=1)
170 DATA 00071F3F3F7F7F7F,00E0F8FCFCFEFEF
E,FEFEFEFCFCF8E,7F7F7F3F3F1F07,FFFFFFFF
FFFFFFFF
180 FOR I=128 TO 132
190 READ X$
200 CALL CHAR(I,X$)
```

```
210 CALL CHAR(I+8,X$)
220 NEXT I
230 CALL CHAR(150,"0000FC7E3F7EFC")
240 CALL CHAR(152,"00FFFFFFFF")
244 CALL CHAR(64,"3C4202040810001")
250 CALL CHAR(94,"")
260 CALL COLOR(15,5,1)
270 CALL COLOR(16,7,1)
280 S$(1)="      LE MOT LE PLUS LONG"
290 S$(2)="       LE COMPTE EST BON"
300 S$(3)=""
310 GOSUB 26000
320 PRINT "      JEU DES CHIFFRES":::"
  ET DES LETTRES"::::::::::
330 PRINT " tapez toujours une touche"::"
 quelconque pour continuer"
340 CALL HCHAR(14,12,152,8)
341 R=-1
350 GOSUB 27000
500 PRINT S$(1):::::"  au debut vous choi
sissez :":"v-voyelles   c-consonnes."::
:
520 PRINT "  chaque joueur, lorsque sonno
m est souligne, indique lenombre de let
tres du mot":"qu'il a trouve."
530 PRINT ::"  le joueur dont le nom est
souligne doit inscrire son  mot.":::"
validation : verifiez dans"
534 PRINT "le dictionnaire si le mot   es
t acceptable et repondez :o-oui    n-non
";
540 CALL HCHAR(3,12,150,10)
550 GOSUB 27000
630 PRINT S$(2):::::"  a l'aide des 4 ope
rations,essayez de parvenir au":"nombre
 encadre en utilisant"
640 PRINT "tout ou partie des 6 nombresou
 chiffres proposes.":::
650 PRINT "  chaque joueur, lorsque sonno
m est souligne, indique lenombre le plu
s proche qu'il a trouve.":::
652 PRINT "  le joueur designe doit     do
nner sa solution.":::"  symboles utilis
ables :"
656 PRINT "  +  -  x *   : /   =";
660 CALL HCHAR(3,12,152,10)
670 GOSUB 27000
1020 R=0
1021 GOSUB 26000
1030 PRINT "CHOISISSEZ LE JEU QUI VOUS  C
ONVIENT :"::"         1 le mot le plus lo
ng"::
1040 PRINT "        2 le compte est bon"::
"     3 des chiffres":TAB(15);"et des
 lettres":::
1050 PRINT "COMBIEN DE JOUEURS ETES-VOUS"
::TAB(13);"un":TAB(13);"deux":TAB(13);"
trois":TAB(13);"quatre":::
1060 PRINT "LONGUEUR DU JEU :"::TAB(12);"
-   tours"::TAB(12);"-   tours";
1070 GOSUB 27000
1080 IF (K>48)*(K<52)THEN 1110
1090 GOSUB 28000
1100 GOTO 1080
1110 CALL SOUND(200,-5,5)
1120 JEU=K-48
1130 CALL HCHAR(2+2*JEU,10,150)
1140 FOR I=1 TO 4
1150 CALL HCHAR(13+I,13,48+I)
1155 A(I)=0
1160 NEXT I
1170 GOSUB 28000
1180 IF (K<49)+(K>52)THEN 1170
1190 CALL SOUND(200,-5,5)
1200 QJ=K-48
1210 CALL HCHAR(13+QJ,13,150)
1215 JRM=0
1220 FOR I=1 TO 2
1230 CALL HCHAR(20+2*I,15,48+I)
1240 CALL HCHAR(20+2*I,16,48-(QJ>2)*2*I)
1250 CALL HCHAR(20+2*I,13,48+I)
1260 NEXT I
1270 GOSUB 28000
1280 IF (K<49)+(K>50)THEN 1270
1290 CALL SOUND(200,-5,5)
1300 TOUR=(10-(QJ>2)*2)*(K-48)
1310 CALL HCHAR(20+2*(K-48),13,150)
1320 FOR I=1 TO 500
1321 NEXT I
1330 GOSUB 26000
1340 IF JEU<3 THEN 1360
1350 PRINT "      JEU DES CHIFFRES":::"
  ET DES LETTRES"
1360 PRINT S$(JEU):::::::
1370 PRINT "veuillez inscrire vos noms"::
"ou vos prenoms :":::
1430 S=9
1434 CALL HCHAR(15,12,150,10)
1440 GOSUB 27000
1450 FOR I=1 TO QJ
1454 PRINT
1460 INPUT S$(I)
```

```
1470 S$(I)=SEG$(S$(I),1,13)
1480 SC(I)=0
1490 NEXT I
1500 CALL SOUND(300,800,6)
1510 CALL CLEAR
1520 FOR I=5 TO 8
1530 CALL COLOR(I,2,8)
1540 NEXT I
1542 PRINT TAB(25);"tour"::TAB(26);"00"::
1543 CALL HCHAR(21,27,152,4)
1550 FOR J=1 TO QJ
1560 X=INT(QJ*RND)+1
1570 IF S$(X)="" THEN 1560
1580 LG(J)=LEN(S$(X))
1590 FOR W=1 TO LG(J)
1600 K=ASC(SEG$(S$(X),W,1))
1610 CALL HCHAR(V-1,H+W-1,K+(K<95)*(K>64)
*32)
1620 NEXT W
1630 S$(X)=""
1640 CALL HCHAR(V+1,H+2,48,3)
1650 NEXT J
1700 FOR TOUR=1 TO TOUR
1710 IF JEU=1 THEN 1760
1720 GOSUB 10000
1740 IF JEU=2 THEN 3500
1750 TOUR=TOUR+1
1760 GOSUB 5000
1780 IF (JEU=1)+(QJ=3)THEN 3500
1790 TOUR=TOUR+1
1800 GOSUB 5000
1820 IF (QJ=4)-(TOUR<>9)*(TOUR<>19)THEN 3
500
1830 TOUR=TOUR+1
1840 GOSUB 10000
3500 NEXT TOUR
3510 S=-5
3520 GOSUB 20000
3530 CALL SCREEN(8)
3540 FOR I=1 TO 13
3550 CALL HCHAR(20,7+I,ASC(SEG$("r.recomm
encer",I,1)))
3560 CALL HCHAR(22,7+I,ASC(SEG$("f.finir
    ",I,1)))
3570 NEXT I
3580 GOSUB 28000
3590 IF K=82 THEN 280
3600 IF K<>70 THEN 3580
3610 CALL CLEAR
3620 END
5000 GOSUB 20000
```

```
5010 FOR I=12 TO 14
5020 CALL HCHAR(I,5,94,21)
5030 NEXT I
5040 CALL HCHAR(21,17,118)
5050 CALL HCHAR(21,18,47)
5060 CALL HCHAR(21,19,99)
5070 CALL HCHAR(21,15,150)
5080 FOR I=1 TO 9
5090 CALL SOUND(80,2000,4)
5100 FOR W=1 TO DL/10
5110 CALL HCHAR(21,22,63)
5120 CALL KEY(3,K,S)
5124 IF S=-1 THEN 5150
5130 IF K=86 THEN 5190
5140 IF K=67 THEN 5260
5150 CALL HCHAR(21,22,32)
5160 NEXT W
5170 CALL SOUND(-100,150,5)
5180 IF INT(2*RND)THEN 5260
5190 CALL SOUND(200,800,6,-8,6)
5200 CALL HCHAR(21,22,118)
5210 IF INT(45*RND)THEN 5240
5220 X$="Y"
5230 GOTO 5290
5240 X$=SEG$("AAAEEEEEIIOOUU",INT(14*RND)
+1,1)
5250 GOTO 5290
5260 CALL SOUND(200,300,6,-4,6)
5270 CALL HCHAR(21,22,99)
5280 X$=SEG$("BBBCCCCDDDFFGGHHJKLLLLLMMMN
NNNNPPPQRRRRRRRSSSSSSSTTTTTTVWXZ",INT(5
9*RND)+1,1)
5290 CALL HCHAR(13,5+2*I,ASC(X$))
5300 Z$=Z$&X$
5310 NEXT I
5312 CALL HCHAR(21,15,32,8)
5314 CALL SOUND(400,660,2)
5320 GOSUB 25000
5330 FOR I=1 TO 7
5340 CALL HCHAR(21,6+I,ASC(SEG$("combien"
,I,1)))
5350 NEXT I
5360 CALL HCHAR(21,5,150)
5370 FOR I=1 TO QJ
5380 CALL HCHAR(V,H,150,LG(J))
5390 CALL SOUND(150,250*J,4)
5400 FOR W=1 TO DL/5
5410 CALL HCHAR(21,16,63)
5420 CALL KEY(3,K,S)
5430 IF (K>47)*(K<58)THEN 5500
5440 CALL HCHAR(21,16,32)
```

```
5450 NEXT W
5460 CALL SOUND(150,150,4)
5470 CALL HCHAR(21,16,48)
5480 A(J)=48
5490 GOTO 5540
5500 CALL SOUND(300,600,3)
5510 CALL HCHAR(21,16,K)
5520 A(J)=K
5530 IF K=57 THEN 5620
5540 CALL HCHAR(V,H,32,15)
5550 J=-J*(J<QJ)+1
5560 NEXT I
5570 GOTO 5590
5580 J=-J*(J<QJ)+1
5590 FOR I=1 TO QJ
5600 IF A(J)<A(I)THEN 5580
5610 NEXT I
5620 CALL HCHAR(21,5,32)
5621 CALL HCHAR(V,H,150,LG(J))
5630 CALL HCHAR(21,16,A(J))
5640 CALL SOUND(500,250*J,2)
5642 FOR I=16 TO 18
5643 CALL HCHAR(I,5,94,21)
5644 NEXT I
5650 W=0
5654 X$=Z$
5660 FOR I=1 TO A(J)-48
5670 CALL HCHAR(17,5+I*2,64)
5680 CALL KEY(3,K,S)
5684 IF S=-1 THEN 5700
5690 IF (K>64)*(K<91)THEN 5750
5700 CALL HCHAR(17,5+I*2,94)
5710 W=W+1
5720 IF W<DL THEN 5670
5730 GOSUB 29000
5740 GOTO 7050
5750 CALL SOUND(100,-5,4)
5754 CALL HCHAR(17,5+I*2,K)
5760 X=POS(Z$,CHR$(K),1)
5762 IF X=0 THEN 7000
5763 R=POS(X$,CHR$(K),1)
5764 CALL HCHAR(13,5+R*2,94)
5765 X$=SEG$(X$,1,R-1)&"?"&SEG$(X$,R+1,9)
5770 Z$=SEG$(Z$,1,X-1)&SEG$(Z$,X+1,9)
5780 NEXT I
5790 FOR I=1 TO 15
5800 CALL HCHAR(23,6+I,ASC(SEG$("validati
on(o,n)",I,1)))
5810 NEXT I
5820 CALL SOUND(150,250*J,4)
5830 CALL HCHAR(23,5,150)
5840 CALL HCHAR(23,24,63)
5850 CALL KEY(3,K,S)
5860 CALL HCHAR(23,24,32)
5870 IF S=0 THEN 5840
5880 CALL SOUND(-200,-5,4)
5890 CALL HCHAR(23,24,K+32)
5900 CALL HCHAR(23,5,32)
5910 IF K=78 THEN 7050
5920 IF K=79 THEN 5940
5930 GOTO 5820
5940 CALL SOUND(300,800,3)
5950 X=A(J)
5960 FOR I=1 TO 4
5970 A(I)=0
5980 NEXT I
5990 A(J)=X-48
6000 CALL HCHAR(V+1,H+6,43)
6010 CALL HCHAR(V+1,H+7,X)
6020 GOTO 7150
7000 CALL SOUND(300,150,3)
7020 FOR I=1 TO 18
7030 CALL HCHAR(23,4+I,ASC(SEG$("erreur d
e lettre !",I,1)))
7040 NEXT I
7050 CALL SOUND(300,150,3)
7060 X=A(J)
7070 A(J)=0
7080 K=J
7090 FOR J=1 TO QJ
7100 IF J=K THEN 7140
7110 A(J)=X-48
7120 CALL HCHAR(V+1,H+6,43)
7130 CALL HCHAR(V+1,H+7,X)
7140 NEXT J
7150 CALL SOUND(100,2000,4)
7160 GOSUB 28000
7180 CALL SOUND(200,800,4)
7190 RETURN
10000 GOSUB 20000
10002 FOR I=1 TO 21
10004 CALL HCHAR(24,4+I,ASC(SEG$("tirage
des chiffres !",I,1)))
10006 NEXT I
10100 X=N(Z)
10110 Z$=STR$(X)
10120 S$(5)=""
10130 FOR I=1 TO 5
10140 S=N(Z)
10150 ON INT(4*RND)+1 GOTO 10170,10210,10
250,10290
10170 W=X+S
```

```
10180 IF (W<1)+(W>999)THEN 10140
10190 S$(I)=STR$(X)&"+"
10200 GOTO 10570
10210 W=X-S
10220 IF (W<1)+(W>999)THEN 10140
10230 S$(I)=STR$(X)&"-"
10240 GOTO 10570
10250 W=X*S
10260 IF (W<1)+(W>999)+(W=X)+(W=S)THEN 10
140
10270 S$(I)=STR$(X)&"x"
10280 GOTO 10570
10290 IF I=5 THEN 10575
10300 K=N(Z)
10310 ON INT((Z+1)/2)GOTO 10320,10360,104
00
10320 W=K+S
10330 IF (W<1)+(W>300)THEN 10300
10340 S$(I)=STR$(K)&"+"
10350 GOTO 10410
10360 W=K-S
10370 IF (W<1)+(W>300)THEN 10300
10380 S$(I)=STR$(K)&"-"
10390 GOTO 10410
10400 W=K*S
10402 IF (W<1)+(W>300)+(W=K)+(W=S)THEN 10
300
10404 S$(I)=STR$(K)&"x"
10410 S$(I)=S$(I)&STR$(S)&"="&STR$(W)
10415 I=I+1
10430 Z$=STR$(K)&" "&Z$&" "&STR$(S)
10435 K=W
10440 ON INT((Z+1)/2)GOTO 10450,10490,105
30
10450 W=X+K
10460 IF K>999 THEN 10440
10470 S$(I)=STR$(X)&"+"
10480 GOTO 10555
10490 W=X-K
10500 IF K<1 THEN 10440
10510 S$(I)=STR$(X)&"-"
10520 GOTO 10555
10530 W=X*K
10540 IF (K>999)+(W=X)+(W=K)THEN 10440
10550 S$(I)=STR$(X)&"x"
10555 S$(I)=S$(I)&STR$(K)&"="&STR$(W)
10560 GOTO 10610
10570 S$(I)=S$(I)&STR$(S)&"="&STR$(W)
10575 IF INT(2*RND)THEN 10600
10580 Z$=STR$(S)&" "&Z$
10590 GOTO 10610
10600 Z$=Z$&" "&STR$(S)
10610 X=W
10620 NEXT I
10630 IF (X<101)+(X>999)THEN 10100
10631 CALL COLOR(13,1,1)
10632 CALL HCHAR(11,24,132,5)
10633 CALL HCHAR(15,24,132,5)
10634 CALL VCHAR(12,23,132,3)
10635 CALL VCHAR(12,29,132,3)
10636 CALL HCHAR(11,23,128)
10637 CALL HCHAR(11,29,129)
10638 CALL HCHAR(15,29,130)
10639 CALL HCHAR(15,23,131)
10640 CALL COLOR(13,13,1)
10650 X$=STR$(X)
10654 CALL HCHAR(13,25,48,3)
10656 CALL HCHAR(24,1,32,32)
10660 FOR I=3 TO 1 STEP -1
10670 FOR W=48 TO ASC(SEG$(X$,I,1))
10680 CALL SOUND(50,5000,3)
10690 CALL HCHAR(13,24+I,W)
10700 NEXT W
10710 NEXT I
10720 FOR I=1 TO LEN(Z$)
10730 CALL HCHAR(11,4+I,ASC(SEG$(Z$,I,1))
)
10740 NEXT I
10750 GOSUB 25000
10760 FOR I=1 TO 7
10770 CALL HCHAR(24,11+I,ASC(SEG$("combie
n",I,1)))
10780 NEXT I
10790 FOR I=1 TO QJ
10800 CALL HCHAR(V,H,150,LG(J))
10810 X$=""
10820 R=0
10830 CALL SOUND(150,250*J,4)
10840 CALL HCHAR(24,10,150)
10850 FOR W=1 TO 4
10870 CALL HCHAR(24,20+W,32)
10880 R=R+1
10890 IF R>DL/3 THEN 11010
10900 CALL HCHAR(24,20+W,63)
10910 CALL KEY(3,K,S)
10920 IF (K<48)+(K>57)+(S=-1)THEN 10870
10924 CALL SOUND(80,800,4)
10930 CALL HCHAR(24,20+W,K)
10940 X$=X$&STR$(K-48)
10950 IF (W=3)*(X>900)*(X$<"150")THEN 109
80
10960 IF W=3 THEN 10990
```

```
10980 NEXT W
10990 A(J)=VAL(X$)
11000 GOTO 11050
11010 IF (X<200)*(X$>"70")THEN 10990
11020 CALL SOUND(150,150,4)
11030 CALL HCHAR(24,21,48,3)
11040 A(J)=0
11050 CALL HCHAR(24,10,32)
11060 CALL HCHAR(V,H,32,15)
11070 IF A(J)=X THEN 11160
11080 J=-J*(J<QJ)+1
11090 CALL HCHAR(24,21,32,4)
11100 NEXT I
11110 GOTO 11130
11120 J=-J*(J<QJ)+1
11130 FOR I=1 TO QJ
11140 IF ABS(X-A(J))>ABS(X-A(I))THEN 1112
0
11150 NEXT I
11160 FOR I=16 TO 18
11170 CALL HCHAR(I,10,32,4)
11180 NEXT I
11185 CALL HCHAR(V,H,150,LG(J))
11190 IF A(J)=0 THEN 11950
11200 X$=STR$(A(J))
11210 X=6-(X=A(J))*2
11220 FOR I=1 TO LEN(X$)
11230 CALL HCHAR(24,20+I,ASC(SEG$(X$,I,1)
))
11240 NEXT I
11250 CALL SOUND(500,200*J,2)
11260 A(4)=A(J)
11270 A(1)=0
11280 I=14
11290 R=0
11300 W=4
11310 A(3)=1
11320 X$=""
11330 FOR Y=A(3)TO 5
11340 W=W+1
11350 CALL HCHAR(I,W,63)
11360 A(1)=A(1)+1
11380 CALL KEY(3,K,S)
11390 CALL HCHAR(I,W,32)
11392 IF A(1)<DL*3/2 THEN 11400
11393 GOSUB 29000
11394 GOTO 11790
11400 IF S<>1 THEN 11350
11410 CALL SOUND(80,800,4)
11420 IF (K<48)+(K>57)THEN 11500
11425 CALL HCHAR(I,W,K)
11430 X$=X$&STR$(K-48)
11440 NEXT Y
11450 IF A(2)<>VAL(X$)THEN 11740
11460 IF A(2)=A(4)THEN 11870
11470 I=I+2
11480 Z$=Z$&X$
11490 GOTO 11290
11500 IF Y=A(3)THEN 11330
11505 S=POS(Z$,X$,1)
11510 IF S=0 THEN 11740
11520 Z$=SEG$(Z$,1,S-1)&SEG$(Z$,S+LEN(X$)
,50)
11530 IF R<>0 THEN 11560
11540 A(2)=VAL(X$)
11550 GOTO 11670
11560 IF R<>43 THEN 11590
11570 A(2)=A(2)+VAL(X$)
11580 GOTO 11670
11590 IF R<>45 THEN 11620
11600 A(2)=A(2)-VAL(X$)
11610 GOTO 11670
11620 IF (R<>42)*(R<>120)THEN 11650
11630 A(2)=A(2)*VAL(X$)
11640 GOTO 11670
11650 IF (R<>47)*(R<>58)THEN 11740
11660 A(2)=A(2)/VAL(X$)
11670 IF W<18 THEN 11700
11680 W=5
11690 I=I+2
11700 R=K-(K=88)*32
11710 CALL HCHAR(I,W,R)
11720 A(3)=6-LEN(STR$(A(2)))
11730 IF K=61 THEN 11320 ELSE 11310
11740 CALL SOUND(500,150,2)
11750 I=I+2+(I=22)
11760 FOR W=1 TO 9
11770 CALL HCHAR(I,7+W,ASC(SEG$(" erreur
!",W,1)))
11780 NEXT W
11790 FOR W=1 TO 4
11791 A(W)=0
11792 NEXT W
11793 W=J
11795 FOR J=1 TO QJ
11800 IF W=J THEN 11850
11810 A(J)=X
11820 CALL HCHAR(V+1,H+6,43)
11830 CALL HCHAR(V+1,H+7,48+X)
11850 NEXT J
11860 GOTO 11950
11870 CALL SOUND(500,600,2)
```

```
11880 FOR W=1 TO 4
11890 A(W)=0
11900 NEXT W
11905 A(J)=X
11910 CALL HCHAR(V+1,H+6,43)
11920 CALL HCHAR(V+1,H+7,48+X)
11930 IF X=8 THEN 11969
11950 FOR I=1 TO DL*4
11955 NEXT I
11960 FOR I=14 TO 22
11961 CALL HCHAR(I,5,32,18)
11962 NEXT I
11964 FOR I=1 TO 5
11965 FOR W=1 TO LEN(S$(I))
11966 CALL HCHAR(12+I*2,4+W,ASC(SEG$(S$(I
),W,1)))
11967 NEXT W
11968 NEXT I
11969 CALL SOUND(100,2000,4)
11970 GOSUB 28000
11980 CALL SOUND(200,800,4)
11990 RETURN
20000 CALL HCHAR(2,1,32,32)
20010 CALL HCHAR(7,1,32,32)
20020 CALL HCHAR(11,1,32,313)
20030 FOR I=21 TO 24
20040 CALL HCHAR(I,1,32,25)
20050 NEXT I
20060 JRM=-JRM*(JRM(QJ)+1
20070 CALL SCREEN(VAL(SEG$("11140415",JRM
*2-1,2)))
20080 FOR J=1 TO QJ
20100 SC(J)=SC(J)+A(J)
20110 X$=STR$(SC(J))
20120 X=LEN(X$)
20130 CALL HCHAR(V+1,H+6,32,2)
20132 IF A(J)=0 THEN 20170
20140 FOR I=X TO 1 STEP -1
20150 CALL HCHAR(V+1,H+4-X+I,ASC(SEG$(X$,
I,1)))
20160 NEXT I
20170 NEXT J
20180 IF S=-5 THEN 21000
20190 X$=STR$(TOUR)
20200 X=LEN(X$)
20210 FOR I=X TO 1 STEP -1
20220 CALL HCHAR(22,29-X+I,ASC(SEG$(X$,I,
1)))
20230 NEXT I
20240 CALL COLOR(14,1,1)
20250 CALL HCHAR(17,10,140,4)
20260 CALL HCHAR(16,11,140,2)
20270 CALL HCHAR(18,11,140,2)
20280 CALL HCHAR(16,10,136)
20290 CALL HCHAR(16,13,137)
20300 CALL HCHAR(18,13,138)
20310 CALL HCHAR(18,10,139)
20320 CALL HCHAR(17,11,52)
20330 CALL HCHAR(17,12,53)
20340 CALL COLOR(14,10,1)
20350 Z$=""
20360 J=JRM
20370 CALL HCHAR(V,H,150,LG(J))
21000 RETURN
25000 I=52
25010 FOR W=52 TO 48 STEP -1
25020 FOR I=I TO 48 STEP -1
25030 CALL SOUND(70,1000,12)
25040 CALL HCHAR(17,11,W)
25050 CALL HCHAR(17,12,I)
25060 CALL KEY(3,K,S)
25070 IF S THEN 25120
25080 CALL SOUND(650,40000,30)
25090 NEXT I
25100 I=57
25110 NEXT W
25120 RETURN
26000 CALL CLEAR
26010 FOR I=1 TO 12
26020 CALL COLOR(I,1,1)
26030 NEXT I
26040 RETURN
27000 FOR I=5 TO 8
27010 CALL COLOR(I,2,1)
27020 NEXT I
27030 FOR I=1 TO 500-(K=0)*500
27040 NEXT I
27050 FOR I=12 TO 1 STEP -1
27060 CALL COLOR(I,2,1)
27070 NEXT I
27080 IF S=9 THEN 28020
28000 CALL KEY(3,K,S)
28010 IF S<>1 THEN 28000
28014 IF R=-1 THEN 26000
28020 RETURN
29000 CALL SOUND(500,150,2)
29002 CALL HCHAR(24,1,32,32)
29010 FOR K=1 TO 20
29020 CALL HCHAR(24,4+K,ASC(SEG$("reponde
z plus vite !",K,1)))
29030 NEXT K
29040 RETURN
```

A l'heure du Pascal

Gérard Santraille

Ces deux petits programmes Pascal permettent à votre TI-99 de gérer l'heure. Leur emploi est extrêmement simple et ressemble à celui employé par d'autres systèmes d'exploitation (tels MS-DOS ou CPM).

Notre TI ne disposant pas de pile permettant de conserver l'heure il convient d'initialiser l'horloge à chaque session du système. Si vous sauvegardez le programme init-heure sous le nom *IH* et le programme *aff-heure* sous le nom *H* rien de plus simple pour les utiliser :

- revenir au mode commande si l'on ne s'y trouve pas déjà;
- sélectionnez *X* (eXecute) et *IH* à What file?.

Si l'on désire initialiser automatiquement l'heure à chaque session du P-système, il faut sauvegarder le programme *init-heure* sous le nom *SYSTEM.STARTUP* sur la disquette racine (celle qui démarre le système). Le programme affiche *Heure Minute*, auquel on répond en indiquant heure et minute courantes séparées par un espace (et non pas par une virgule comme en Basic). L'emploi de *aff-heure* est encore plus simple. *eXecute H* et l'heure s'affiche, le système vous rendant la main.

Détaillons un peu ces deux programmes.

Init-heure écrit sur deux entiers dans l'une des nombreuses places vides du 'directory' de la disquette (voir à ce sujet le numéro 5 de "99 Magazine") l'heure et la minute auxquelles accédera par la suite le programme *aff-heure*. Avant d'aller plus loin, décrivons la procédure intrinsèque *TIME*.

TIME(h1,h2) retourne en deux entiers non signés (c'est-à-dire dont les 16 bits sont significatifs; un entier signé est écrit sur 15 bits, le premier servant à déterminer le signe, un entier non signé ne peut donc être négatif et varie entre 0 et 2 ^ 16 − 1 à savoir 65535) le nombre de périodes (une période = un 50ème de seconde) écoulées depuis que le système a été démarré.

Le nombre de secondes écoulées correspond donc à (h1 ⋆ 65536 + h2) / 50; à partir de là, on peut faire ce que l'on veut.
La valeur stockée par *init-time* correspond à celle indiquée par l'utilisateur, diminuée de l'heure du système au moment de la manipulation (on peut ainsi initialiser l'heure à n'importe quel moment de la session.

Aff-heure fait l'opération inverse; il additionne (après conversion) l'heure indiquée par le système à celle mémorisée sur la disquette. Une petite explication s'impose :

notons *Hr* l'heure réelle et *Hs* l'heure système au moment de l'exécution de *init-heure*. L'information mémorisée est Hr − Hs. Quand on fait appel à *aff-heure*, il s'est écoulé *dh*; l'heure réelle est Hr + dh que l'on retrouve bien en additionnant l'heure système (qui vaut Hs + dh) à l'heure mémorisée. En effet, Hr + dh = (Hr − Hs) + (Hs + dh).

Remarques :

- la carte P-code n'étant pas sélectée en permanence, une certaine dérive de l'heure peut se produire. Il convient alors de rectifier (à la baisse) la valeur 3000 (50 fois 60) que l'on trouve dans la formule de conversion;
- notez, dans les programmes, l'emploi d'une des instructions les plus puissantes du Pascal : l'accès à un élément d'un type enregistrement par *'enregistrement.élément'*. Notez également l'écriture k=65536.0 pour forcer la constante *k* à être réelle afin de pouvoir faire des calculs sur des valeurs supérieures à 32767 (2 ^ 15 − 1).

■

```
program aff_heure;

const k=65536.0;

type
  octet = 0..255;

  bloc1 = record
    tampon1: packed array[1..22] of octet;
    heure  : integer;
    minute : integer;
    tampon2: packed array[1..486] of octet
           end;

var hs1,hs2,h,m:integer;
    rs1,rs2:real;
    b:bloc1;


begin
page(output);
unitread(4,b,512,2);
h:=b.heure;
m:=b.minute;
time(hs1,hs2);
rs1:=hs1;
rs2:=hs2;
if rs2<0 then rs2:=rs2+k;
hs2:=trunc((rs1*k+rs2)/3000);
hs1:=hs2 div 60;
hs2:=hs2 mod 60;
hs2:=hs2+m;
if hs2>=60 then
  begin
  hs2:=hs2-60;
  hs1:=hs1+1;
  end;
hs1:=hs1+h;
hs1:=hs1 mod 24;
gotoxy(0,1);
writeln(hs1:2,'h',hs2:2,'mn')
end.
```

```
program init_heure;

const k=65536.0;

type
  octet = 0..255;

  bloc1 = record
    tampon1: packed array[1..22] of octet;
    heure  : integer;
    minute : integer;
    tampon2: packed array[1..486] of octet
         end;

var h,m,hs1,hs2:integer;
    rs1,rs2:real;
    b:bloc1;


begin
page(output);
unitread(4,b,512,2);
write('Heure Minute : ',chr(7));
readln(h,m);
time(hs1,hs2);
rs1:=hs1;
rs2:=hs2;
if rs2<0 then rs2:=rs2+K;
hs2:=trunc((rs1*K+rs2)/3000);
hs1:=hs2 div 60;
hs2:=hs2 mod 60;
h:=h-hs1;
if h<0 then h:=24+h;
m:=m-hs2;
if m<0 then
  begin
  h:=h-1;
  m:=60+m
  end;
b.heure:=h;
b.minute:=m;
unitwrite(4,b,512,2)
end.
```

Kong

François Blanchard

Ce programme de jeu écrit en **Basic étendu** est une adaptation très libre du célèbre "Donkey Kong".

Scénario

Le but du jeu est d'ammener Robert au sommet de l'échafaudage afin qu'il puisse délivrer sa fiancée, prisonnière de l'ignoble Kong. Robert doit éviter les flammes qui sillonent les étages, en sautant par dessus.

Si vous parvenez au sommet de l'échafaudage cinq fois de suite, vous obtenez un bonus de 1000 points, mais ne vous faites pas d'illusions; les flammes accélèrent.

Utilisation

Seule la manette de jeux numéro deux est utilisée pour les déplacements de Robert et ses sauts au dessus des flammes. Notre ami subit certaines contraintes :

- il ne peut pas sortir de l'écran;
- il ne peut monter qu'aux échelles;
- il ne peut descendre qu'aux échelles.

■

```
100 ! ********************
110 ! *     K O N G      *
120 ! ********************
125 ! *  BASIC ETENDU    *
130 ! ********************
135 !
150 BONUS=0 :: VI=3 :: SC=0 :: V1=-12 ::
V2=14 :: V3=-16 :: V4=18
160 DISPLAY AT(2,1)ERASE ALL:"VOULEZ-VOUS
 AIDER ROBERT A      DELIVRER SA FIANCEE
                     PRISONNIERE         DE
 L'ABOMINABLE" :: DISPLAY AT(13,10):"K
O N G"
170 GOSUB 860
180 CALL KEY(0,K,S):: IF S=0 THEN 180
190 CALL CLEAR :: CALL CHAR(112,"FFFF1824
4281FFFF",59,"8181FF8181FF8181")!POUTRE
,ECHELLE
200 CALL CHAR(104,"101429345E7B361C000000
00000000000000000000000000000")!FLAM
```

```
ME
210 CALL CHAR(120,"3068702070D8D8C8F87060
606060607000000000000000000000000")!MA
RIO D
220 CALL CHAR(124,"000042181818FFFF18185A
1818181818000000000000000000000000")! CR
OIX
230 CALL CHAR(128,"60B0702070D8D8B8F87030
303030307000000000000000000000000")!MA
RIO G
240 CALL CHAR(132,"0C131313000A0E0E1F3F3F
6A55AA00000000A0C020000000000808040A00
000")!FIANCEE 1
250 CALL CHAR(136,"000C040C04354020000000
0000110A33000000000080408000000000000
000")!FIANCEE 2
260 CALL CHAR(114,"006CFEFE7C7C3810")! CO
EUR FIANCEE
270 CALL CHAR(96,"030108080460F0F8783C0E1
E3E786000C080101020060F1F1E3C70787C1E06
00")!KONG 1
280 CALL CHAR(108,"000615371B1E0F05030101
01000000E00060A8ECD838F0A0C010101000000
007")!KONG 2
290 CALL CHAR(100,"60C0607030100000000000
000000000000000000000000000000000000
000")!BRAS GAUCHE KONG
300 CALL CHAR(92,"603060E0C04000000000000
000000000000000000000000000000000000
000")!BRAS D KONG
310 CALL SCREEN(2)
320 FOR L=23 TO 11 STEP -4 :: CALL HCHAR(
L,1,112,32):: NEXT L :: CALL VCHAR(20,3
0,59,3):: CALL VCHAR(16,5,59,3):: CALL VC
HAR(12,18,59,3)
330 CALL VCHAR(8,10,59,3):: CALL VCHAR(5,
25,59,6):: CALL VCHAR(5,27,59,6):: CALL
 HCHAR(7,9,112,16):: CALL HCHAR(4,25,112,
3)
340 CALL COLOR(2,16,1,11,10,1,4,16,1,3,16
,1,4,16,1,5,16,1,6,16,1,7,16,1,8,16,1)
350 CALL SPRITE(#25,132,5,9,200,#26,136,1
4,9,200)
360 DISPLAY AT(3,2):"1UP 0000" :: DISPLAY
 AT(5,2)SIZE(4):"000" :: DISPLAY AT(2,2
8):"3"
370 IF MSC<>0 THEN DISPLAY AT(3,6):MSC
380 CALL SPRITE(#27,96,7,33,124,#28,108,1
1,33,124)
390 CALL SPRITE(#4,104,12,73,1,0,V4,#3,10
4,12,105,1,0,V3,#2,104,12,137,1,0,V2,#1
```

```
,104,12,169,249,0,V1)
400 !  RE DE CHAUSSEE
410 X=161 :: Y=17
420 CALL SPRITE(#8,120,14,X,Y):: CALL MAG
NIFY(3)
430 CALL COINC(#8,#1,8,C):: IF C THEN 730
440 CALL KEY(2,S,K):: IF K THEN IF D THEN
 Y=Y-16 :: CALL M(Y):: CALL LOCATE(#8,1
52,Y):: CALL LOCATE(#8,X,Y)ELSE Y=Y+16 ::
 CALL M(Y):: CALL LOCATE(#8,152,Y):: CA
LL LOCATE(#8,X,Y)
450 CALL JOYST(2,YY,XX):: IF YY=4 THEN Y=
Y+8 :: D=0 :: CALL M(Y):: CALL SPRITE(#
8,120,14,X,Y):: GOTO 430
460 IF YY=-4 THEN Y=Y-8 :: D=1 :: CALL M(
Y):: CALL SPRITE(#8,128,14,X,Y):: GOTO
430
470 IF XX=4 THEN IF Y=233 THEN X=129 :: C
ALL LOCATE(#8,X,Y):: SC=SC+20 :: GOSUB
760 :: GOTO 500
480 GOTO 430
490 ! 1ere ETAGE
500 CALL COINC(#8,#2,8,C):: IF C THEN 730
510 CALL KEY(2,S,K):: IF K THEN IF D THEN
 Y=Y-16 :: CALL M(Y):: CALL LOCATE(#8,1
21,Y):: CALL LOCATE(#8,X,Y)ELSE Y=Y+16 ::
 CALL M(Y):: CALL LOCATE(#8,121,Y):: CA
LL LOCATE(#8,X,Y)
520 CALL JOYST(2,YY,XX):: IF YY=4 THEN Y=
Y+8 :: D=0 :: CALL M(Y):: CALL SPRITE(#
8,120,14,X,Y):: GOTO 500
530 IF YY=-4 THEN Y=Y-8 :: D=1 :: CALL M(
Y):: CALL SPRITE(#8,128,14,X,Y):: GOTO
540
540 IF XX=4 THEN IF Y=33 THEN X=97 :: CAL
L LOCATE(#8,X,Y):: SC=SC+50 :: GOSUB 76
0 :: GOTO 580
550 IF XX=-4 THEN IF Y=233 THEN X=161 ::
CALL LOCATE(#8,X,Y):: GOTO 430
560 GOTO 500
570 ! 2eme ETAGE
580 CALL COINC(#8,#3,8,C):: IF C THEN 730
590 CALL KEY(2,S,K):: IF K THEN IF D THEN
 Y=Y-16 :: CALL M(Y):: CALL LOCATE(#8,8
9,Y):: CALL LOCATE(#8,X,Y)ELSE Y=Y+16 ::
CALL M(Y):: CALL LOCATE(#8,89,Y):: CALL
 LOCATE(#8,X,Y)
600 CALL JOYST(2,YY,XX):: IF YY=4 THEN Y=
Y+8 :: D=0 :: CALL M(Y):: CALL SPRITE(#
8,120,14,X,Y):: GOTO 580
610 IF YY=-4 THEN Y=Y-8 :: D=1 :: CALL M(
```

```
Y):: CALL SPRITE(#8,128,14,X,Y):: GOTO
580
620 IF XX=4 THEN IF Y=137 THEN X=65 :: CA
LL LOCATE(#8,X,Y):: SC=SC+70 :: GOSUB 7
60 :: GOTO 660
630 IF XX=-4 THEN IF Y=33 THEN X=129 :: C
ALL LOCATE(#8,X,Y):: GOTO 500
640 GOTO 580
650 ! 3eme etage
660 CALL COINC(#8,#4,9,C):: IF C THEN 730
670 CALL KEY(2,S,K):: IF K THEN IF D THEN
 Y=Y-16 :: CALL M(Y):: CALL LOCATE(#8,5
7,Y):: CALL LOCATE(#8,X,Y)ELSE Y=Y+16 ::
CALL M(Y):: CALL LOCATE(#8,57,Y):: CALL
 LOCATE(#8,X,Y)
680 CALL JOYST(2,YY,XX):: IF YY=4 THEN Y=
Y+8 :: D=0 :: CALL M(Y):: CALL SPRITE(#
8,120,14,X,Y):: GOTO 660
690 IF YY=-4 THEN Y=Y-8 :: D=1 :: CALL M(
Y):: CALL SPRITE(#8,128,14,X,Y):: GOTO
660
700 IF XX=4 THEN IF Y=73 OR Y=193 OR Y=20
9 THEN SC=SC+120 :: CALL SPRITE(#8,120,
14,33,88):: GOSUB 800 :: GOSUB 760 :: GOT
O 390
710 IF XX=-4 THEN IF Y=137 THEN X=97 :: C
ALL LOCATE(#8,X,Y):: GOTO 580
720 GOTO 660
730 DUR=30 :: FOR SON=500 TO 110 STEP -50
 :: DUR=DUR-2 :: CALL SOUND(DUR,SON,0):
: CALL PATTERN(#8,124):: NEXT SON
740 FOR I=1 TO 10 :: CALL SPRITE(#10,100,
2,34,121,#11,92,2,34,139):: CALL COLOR(
#10,7,#11,7):: CALL SOUND(-60,110,0):: NE
XT I
750 CALL DELSPRITE(#10,#11,#8,#1,#2,#3,#4
):: GOSUB 760 :: GOSUB 780
760 DISPLAY AT(5,1)BEEP SIZE(8):SC :: IF
SC>MSC THEN MSC=SC :: DISPLAY AT(3,6)SI
ZE(8):MSC :: DISPLAY AT(2,27)SIZE(3):VI
770 RETURN
780 VI=VI-1 :: DISPLAY AT(2,27)SIZE(3):VI
 :: IF VI=0 THEN 790 ELSE 390
790 FOR I=1 TO 20 :: DISPLAY AT(17,10):"*
GAME OVER*" :: CALL SOUND(100,-4,0):: D
ISPLAY AT(17,10):"" :: NEXT I :: CALL DEL
SPRITE(ALL):: GOTO 150
800 FOR U=1 TO 2 :: CALL HCHAR(2,24,114):
: CALL SOUND(300,220,0):: FOR X=220 TO
1020 STEP 50 :: CALL SOUND(20,X,0):: NEXT
 X :: CALL HCHAR(2,24,32):: NEXT U :: B
ONUS=BONUS+1
810 IF BONUS=5 OR BONUS=10 OR BONUS=15 OR
 BONUS=20 THEN VI=VI+1 :: GOSUB 820 ELS
E RETURN
820 GOSUB 860 :: SC=SC+1000 :: FOR R=33 T
O 161 STEP 32 :: CALL LOCATE(#27,R,124,
#28,R,124):: SON=SON-20 :: CALL SOUND(100
,SON,0):: GOSUB 850 :: NEXT R :: GOSUB
840 :: GOSUB 760
830 VI=VI+1 :: DISPLAY AT(2,27):VI :: V1=
-14 :: V2=16 :: V3=-18 :: V4=20 :: GOTO
 380
840 CALL SOUND(-400,-6,0):: GOSUB 850 ::
RETURN
850 FOR P=1 TO 100 :: NEXT P :: RETURN
860 CALL SOUND(100,784,0)
870 CALL SOUND(100,880,0)
880 CALL SOUND(100,784,0)
890 CALL SOUND(100,622,0)
900 CALL SOUND(100,659,0)
910 CALL SOUND(100,523,0)
920 CALL SOUND(100,440,0)
930 CALL SOUND(100,392,0)
940 CALL SOUND(100,311,0)
950 CALL SOUND(100,392,0)
960 CALL SOUND(100,523,0)
970 CALL SOUND(500,262,0)
980 RETURN
990 SUB M(Y)
1000 IF Y<=17 THEN Y=17 :: SUBEXIT
1010 IF Y>=240 THEN Y=241 :: SUBEXIT
1020 SUBEND
```

Mini-Mémoire moins "mini"

Guy Fix

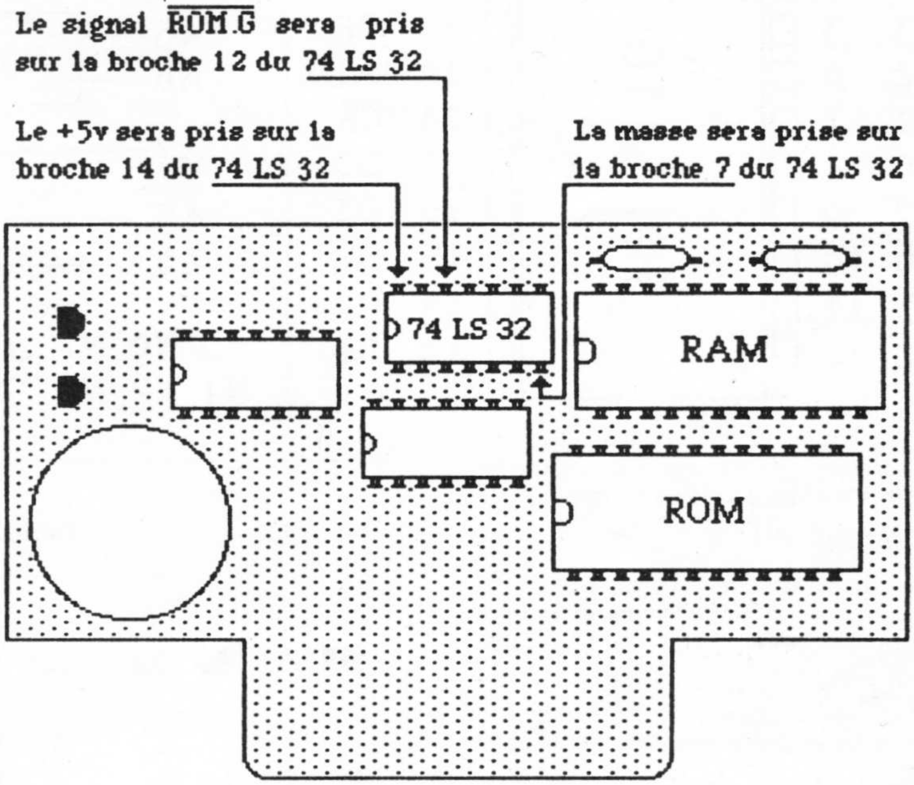
De nombreux possesseurs du TI-99 ont acheté la Mini-Mémoire mais, ne possédant pas d'extension, ils doivent se contenter des quatre Ko disponibles. De plus, les routines nécessaire à la gestion et l'assembleur ligne par ligne laissent peu de place pour la programmation. Cependant, on peut, avec un peu d'astuce, augmenter la capacité de la Mini-Mémoire d'environ 1,5 Ko.

Dans l'espace mémoire réservé à la ROM (mémoire à lecture seule) de la Mini-Mémoire, il existe une zone inutilisé, de l'adresse >697E à l'adresse >6F0E. Il suffit donc d'adresser un boîtier RAM (mémoire à lecture et écriture) dans cette zone. Pour simplifier le circuit de décodage d'adresse, la zone s'étend de >6A00 à >6F00.

Il faut noter que pour tout ce qui suit, il faudra intervenir sur les composants de la Mini-Mémoire; la garantie accordée par Texas Instruments cessera donc dès cet instant.

Examinons maintenant l'intérieur du boîtier; comme d'habitude, on enlève la vis qui se trouve au dos, puis on pousse les deux ergots et on ouvre... Le circuit imprimé apparaît comme sur la figure 1, où seuls les composants principaux sont représentés.

Figure 1



On s'aperçoit que pour des raisons de place, Texas a supperposé les deux boîtier RAM de 2 Ko chacun. Nous allons faire de même et placer un troisième boîtier (mémoire 2 Ko/8 bits HMG 116 LP) au-dessus des deux autres.

Voir figure 2

Il faut donc souder les broches du boîtier que l'on rajoute aux broches des circuits existants déjà sur la carte, en prenant soin de placer le repère de la mémoire *HMG 116* au-dessus du repère du boîtier déjà soudé.

Voir figure 3

Seule la patte *CE* n'est pas soudée; elle va au circuit de décodage d'adresse. Une résistance de 220 ohms va de la patte *CE* au +5v, elle sera intégrée au circuit de décodage.

Circuit de décodage

Un signal *ROM.G*, décodé à l'intérieur de la console, valide les adresses >6000 à >7FFF. Ce signal utilisé conjointement à A3, A4, A5, A6 et A7 permet d'obtenir le tableau 1, et qui représente la configuration du bus d'adresses de >6A00 à 6EFF. La ligne A0 est la ligne de poids fort.

Ce montage peut être effectué sur un circuit imprimé ou sur une plaque d'essai. Les liaisons seront réalisées à l'aide de fil souple entre le boîtier Mini-Mémoire et le circuit de décodage. Le signal *ROM.G* sera pris au niveau du *74 LS 32* implanté d'origine sur la carte, ainsi que le +5v et la masse. Les liaisons des lignes

Figure 2

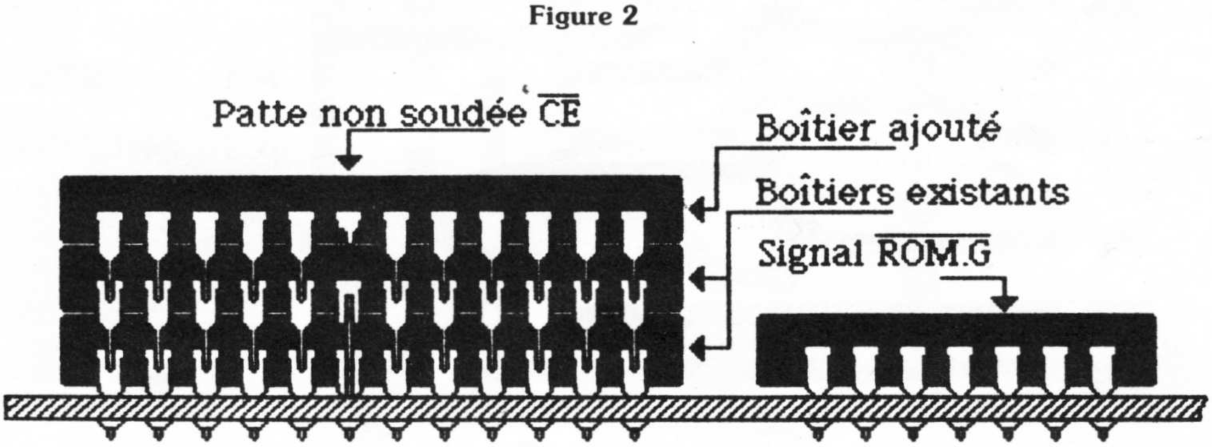
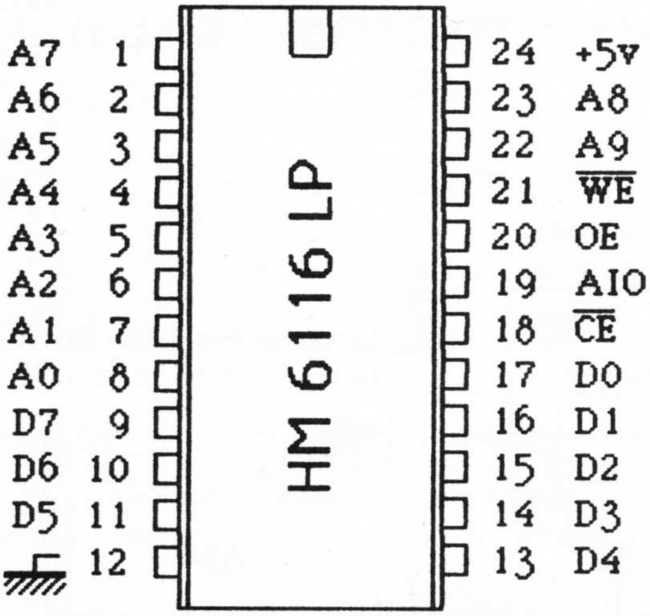
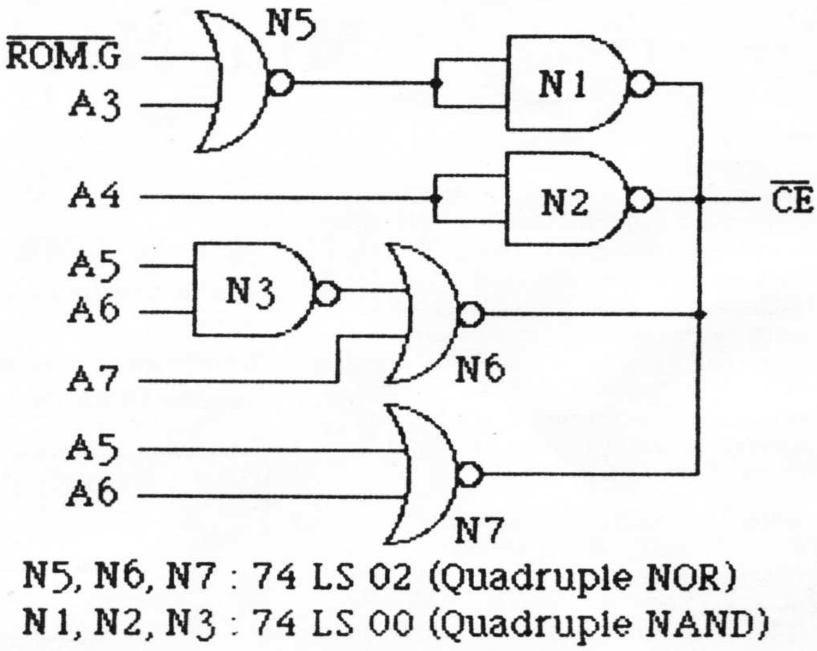


Figure 3



Décodage



d'adresses seront effectuées sur le boîtier mémoire *HM 6116* rajouté.

Voir figure 4

Le circuit ainsi réalisé, on isolera le dessous, côté cuivre, avec un isolant et on placera le circuit imprimé dans le boîtier de la Mini-Mémoire.

Nous pourrons maintenant écrire entre les adresses >6A00 et >6EFF.

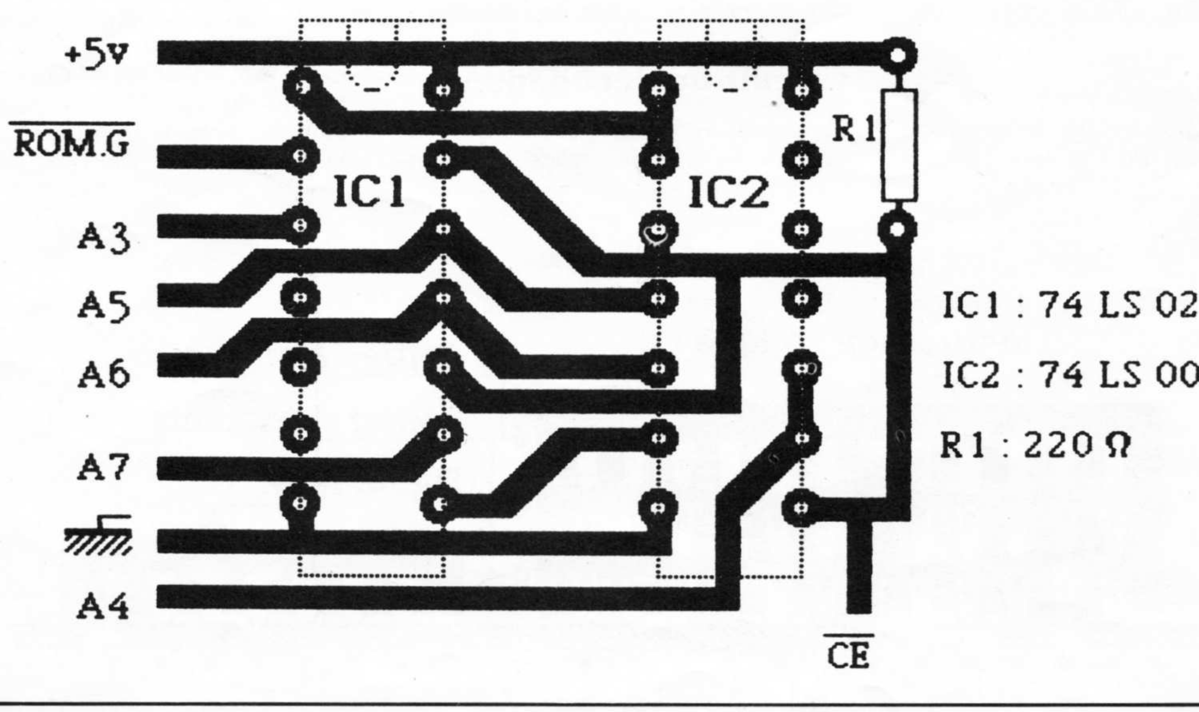
A partir des mêmes circuits, on peut très bien envisager d'étendre la mémoire du TI; pour cela, il faut connaître le brochage du connecteur d'extension (99 Magazine numéro 5) et savoir que le TMS 9900 effectue la validation des adresses durant la phase active du signal *MEMEN*. ■

Tableau 1

A0	A1	A2	A3	A4	A5	A6	A7				
0	1	1	0	1	0	1	0	»	>6A00	à	>6AFF
0	1	1	0	1	0	1	1	»	>6B00	à	>6BFF
0	1	1	0	1	1	0	0	»	>6C00	à	>6CFF
0	1	1	0	1	1	0	1	»	>6D00	à	>6DFF
0	1	1	0	1	1	1	0	»	>6E00	à	>6EFF

Décodé par ROM.G

Figure 4



Edipage

Lojik

Ce programme en **Basic TI** permet l'édition de texte par page d'écran, sur 21 lignes et 28 colonnes. Les textes, conservés sur des cassettes magnétiques, sont récupérés par page ou partie de page. Un ensemble de fonctions permet une manipulation facile du texte.

Tous les caractères du clavier standard sont acceptés, à l'exception des touches de contrôle et de fonction réservées (codes ASCII de 1 à 31 et de 129 à 159.

Des touches de fonctions diverses ont été prévues afin de rendre cet éditeur plus souple. Ces touches sont :

ENTER est utilisée pour la fin de l'entrée d'une ligne. Le curseur se positionne au début de la ligne suivante;

FCTN-S déplacement du curseur vers la gauche;

FCTN-D déplacement du curseur vers la droite;

FCTN-X déplacement du curseur vers le bas;

FCTN-E déplacement du curseur vers le haut;

CTRL-P conserve et affiche la position du curseur pour une utilisation future avec *CTRL-M* (3,3 par défaut);

CTRL-M conserve et affiche les limites d'une portion de texte délimitée par la position sauvegardée du curseur (CTRL-P) et la position courante du curseur;

CTRL-T transfère la portion de texte délimitée par *CTRL-M* à la position actuelle du curseur. Le texte contenu entre les limites données est effacé puis affiché à la position courante du curseur;

CTRL-B efface la portion de texte délimitée par *CTRL-M*;

CTRL-C copie une partie du texte contenu dans une autre page sauvegardée sur cassette. Les limites du texte à récupérer sont données par *CTRL-M*;

CTRL-I insère un espace dans une ligne. Le dernier caractère de la ligne doit être un espace;

CTRL-D provoque la destruction du caractère pointé par le curseur. Le texte de la ligne est reporté vers la gauche;

CTRL-E efface une ligne. Les lignes placées sous la ligne courante sont déplacées vers le haut;

CTRL-O fait de la place pour l'insertion d'une ligne en repoussant les lignes suivantes vers le bas. La dernière ligne doit être vide;

CTRL-N est utilisée pour la numérotation des pages. Pressez *CTRL-N* et le curseur ira se positionner après le mot *PAGE* situé en haut de l'écran, puis entrez les deux chiffres identifiant votre page;

CTRL-S effectue la sauvegarde de la page courante sur une cassette;

CTRL-R charge le texte d'une page enregistrée sur une cassette.

■

```
100 REM **********************
110 REM *      EDIPAGE       *
120 REM **********************
130 REM *     BASIC  TI      *
140 REM **********************
150 REM
160 REM ****************
170 REM * INITIALISER  *
180 REM * TABLE DU TEXTE*
190 REM ****************
200 DIM T$(21,28)
210 PAG=1
220 FOR I=1 TO 21
230 FOR J=1 TO 28
240 T$(I,J)=" "
250 NEXT J
260 NEXT I
270 REM ****************
280 REM * AFFICHAGE DU  *
290 REM * CADRAGE ET DE *
300 REM * L'ENTETE      *
310 REM ****************
320 CALL CLEAR
330 CALL COLOR(13,14,16)
340 CALL CHAR(129,"FFFFFFFFFFFFFFFF")
350 CALL CHAR(130,"007E7E7E7E7E7E00")
360 CALL CHAR(131,"00003C3C3C3C0000")
370 CALL CHAR(132,"0000001818000000")
380 CALL CHAR(133,"0000000000000000")
390 CALL SCREEN(8)
400 LA=1
410 CA=3
420 CALL HCHAR(1,1,133,64)
430 CALL VCHAR(1,1,133,48)
440 CALL VCHAR(1,31,133,48)
450 L$=CHR$(132)&CHR$(131)&CHR$(130)&CHR$
(129)&"-EDIPAGE par Lojik-"&CHR$(129)&C
HR$(130)&CHR$(131)&CHR$(132)
460 GOSUB 3550
470 LA=2
480 CA=13
490 L$="Page "&STR$(PAG)
500 GOSUB 3550
510 L=3
520 C=3
530 PL=3
540 PC=3
550 FOR I=1 TO 100
```

```
560 NEXT I
570 CALL HCHAR(24,1,133,32)
580 CALL HCHAR(L,C,129)
590 REM ****************
600 REM * ENTREE DES    *
610 REM * CARACTERES ET*
620 REM * DES FONCTIONS*
630 REM ****************
640 CALL KEY(0,K,S)
650 IF S=0 THEN 640
660 IF (K<32)+(K>127)THEN 800
670 IF S<>1 THEN 640
680 CALL HCHAR(L,C,K)
690 T$(L-2,C-2)=CHR$(K)
700 C=C+1
710 IF C<=30 THEN 640
720 C=3
730 L=L+1
740 IF L<=23 THEN 640
750 L$="FIN DE PAGE"
760 GOSUB 3630
770 L=23
780 C=30
790 GOTO 550
800 CALL HCHAR(L,C,ASC(T$(L-2,C-2)))
810 REM ******************
820 REM * DEPLACEMENT DU *
830 REM * CURSEUR PAR LES*
840 REM * 4 FLECHES      *
850 REM ******************
860 IF K<>8 THEN 920
870 IF C<=3 THEN 900
880 C=C-1
890 GOTO 580
900 C=30
910 GOTO 580
920 IF K<>9 THEN 980
930 IF C>=30 THEN 960
940 C=C+1
950 GOTO 580
960 C=3
970 GOTO 580
980 IF K<>10 THEN 1040
990 IF L>=23 THEN 1020
1000 L=L+1
1010 GOTO 580
1020 L=3
1030 GOTO 580
1040 IF K<>11 THEN 1150
1050 IF L<=3 THEN 1080
1060 L=L-1
1070 GOTO 580
1080 L=23
1090 GOTO 580
1100 REM ****************
1110 REM *FIN D'ENTREE   *
1120 REM *D'UNE LIGNE    *
1130 REM *PAR ENTER      *
1140 REM ****************
1150 IF K<>13 THEN 1250
1160 IF L=23 THEN 580
1170 L=L+1
1180 C=3
1190 GOTO 580
1200 REM ****************
1210 REM *SELECTION DE   *
1220 REM *LA POSITION DU*
1230 REM *CURSEUR        *
1240 REM ****************
1250 IF K<>144 THEN 1370
1260 PL=L
1270 PC=C
1280 L$="COORDONNEES: "&STR$(PL-2)&"-"&ST
R$(PC-2)
1290 GOSUB 3630
1300 GOTO 580
1310 REM ***************
1320 REM *SELECTION DES*
1330 REM *LIMITES D'UNE*
1340 REM *PORTION DE   *
1350 REM *L'ECRAN      *
1360 REM ***************
1370 IF K<>141 THEN 1540
1380 CL=L
1390 CC=C
1400 IF PL<=CL THEN 1430
1410 CL=PL
1420 PL=L
1430 IF PC<=CC THEN 1460
1440 CC=PC
1450 PC=C
1460 L$="LIMITE: "&STR$(PL-2)&"-"&STR$(PC
-2)&" A "&STR$(CL-2)&"-"&STR$(CC-2)
1470 GOSUB 3630
1480 GOTO 580
1490 REM ****************
1500 REM *TRANSFER D'UNE*
1510 REM *PORTION DE    *
1520 REM *MEMOIRE       *
1530 REM ****************
1540 IF K<>148 THEN 1900
1550 IF ((L+(CL-PL))<=23)*((C+(CC-PC))<=3
```

```
0)THEN 1590
1560 L$="ESPACE INSUFFISANT"
1570 GOSUB 3630
1580 GOTO 550
1590 FOR I=PL TO CL
1600 CALL HCHAR(I,PC,32,CC-PC+1)
1610 NEXT I
1620 I2=0
1630 J2=0
1640 FOR I=L TO L+(CL-PL)
1650 FOR J=C TO C+(CC-PC)
1660 CALL HCHAR(I,J,ASC(T$(PL+I2-2,PC+J2-
2)))
1670 J2=J2+1
1680 NEXT J
1690 J2=0
1700 I2=I2+1
1710 NEXT I
1720 FOR I=PL TO CL
1730 FOR J=PC TO CC
1740 CALL GCHAR(I,J,CH)
1750 T$(I-2,J-2)=CHR$(CH)
1760 NEXT J
1770 NEXT I
1780 FOR I=L TO L+(CL-PL)
1790 FOR J=C TO C+(CC-PC)
1800 CALL GCHAR(I,J,CH)
1810 T$(I-2,J-2)=CHR$(CH)
1820 NEXT J
1830 NEXT I
1840 GOTO 580
1850 REM ****************
1860 REM *INSERTION D'UN*
1870 REM *BLANC SUR UNE *
1880 REM *LIGNE         *
1890 REM ****************
1900 IF K<>137 THEN 2060
1910 IF T$(L-2,28)=" " THEN 1950
1920 L$="DEBORDEMENT DE LIGNE"
1930 GOSUB 3630
1940 GOTO 550
1950 FOR I=30 TO C+1 STEP -1
1960 T$(L-2,I-2)=T$(L-2,I-3)
1970 CALL HCHAR(L,I,ASC(T$(L-2,I-2)))
1980 NEXT I
1990 CALL HCHAR(L,C,32)
2000 *SGN CALL  GOTO LOG & !
2010 GOTO 580
2020 REM **************
2030 REM *DESTRUCTION   *
2040 REM *D'UN CARACTERE*
```

```
2050 REM ****************
2060 IF K<>132 THEN 2190
2070 FOR I=C TO 29
2080 T$(L-2,I-2)=T$(L-2,I-1)
2090 CALL HCHAR(L,I,ASC(T$(L-2,I-2)))
2100 NEXT I
2110 CALL HCHAR(L,30,32)
2120 T$(L-2,28)=" "
2130 GOTO 5802140 REM *****************
2150 REM *EFFACER UNE    *
2160 REM *LIGNE ET TASSER*
2170 REM *LES AUTRES     *
2180 REM *****************
2190 IF K<>133 THEN 2360
2200 IF L=23 THEN 2270
2210 FOR I=L TO 22
2220 FOR J=3 TO 30
2230 T$(I-2,J-2)=T$(I-1,J-2)
2240 CALL HCHAR(I,J,ASC(T$(I-2,J-2)))
2250 NEXT J
2260 NEXT I
2270 CALL HCHAR(23,3,32,28)
2280 FOR I=1 TO 28
2290 T$(21,I)=" "
2300 NEXT I
2310 GOTO 580
2320 REM ****************
2330 REM *INSERTION DE  *
2340 REM *LIGNE         *
2350 REM ****************
2360 IF K<>143 THEN 2600
2370 FOR I=1 TO 28
2380 IF T$(21,I)=" " THEN 2420
2390 L$="ESPACE INSUFFISANT"
2400 GOSUB 3630
2410 GOTO 550
2420 NEXT I
2430 IF L=23 THEN 580
2440 FOR I=23 TO L+1 STEP -1
2450 FOR J=3 TO 30
2460 T$(I-2,J-2)=T$(I-3,J-2)
2470 CALL HCHAR(I,J,ASC(T$(I-2,J-2)))
2480 NEXT J
2490 NEXT I
2500 CALL HCHAR(L,3,32,28)
2510 FOR I=1 TO 28
2520 T$(L-2,I)=" "
2530 NEXT I
2540 GOTO 580
2550 REM ****************
2560 REM *MISE A BLANC  *
```

```
2570 REM *D'UNE PORTION *
2580 REM *DE TEXTE      *
2590 REM ****************
2600 IF K<>130 THEN 2710
2610 FOR I=PL TO CL
2620 FOR J=PC TO CC
2630 T$(I-2,J-2)=" "
2640 NEXT J
2650 CALL HCHAR(I,PC,32,CC-PC+1)
2660 NEXT I
2670 GOTO 580
2680 REM ****************
2690 REM *FIN DU PROGRAM*
2700 REM ****************
2710 IF K<>145 THEN 2790
2720 END
2730 REM ***************
2740 REM *SAUVEGARDE   *
2750 REM *D'UNE PAGE   *
2760 REM *D'ECRAN SUR  *
2770 REM *SUR CASSETTE *
2780 REM ***************
2790 IF K<>147 THEN 3000
2800 CALL CLEAR
2810 OPEN #1:"CS1",INTERNAL,OUTPUT,FIXED
192
2820 PRINT #1:PAG
2830 FOR I3=1 TO 19 STEP 6
2840 E$=""
2850 FOR I=I3 TO I3+5
2860 IF I>21 THEN 2910
2870 FOR J=1 TO 28
2880 E$=E$&T$(I,J)
2890 NEXT J
2900 NEXT I
2910 PRINT #1:E$
2920 NEXT I3
2930 CLOSE #1
2940 GOTO 220
2950 REM ****************
2960 REM *RECUPERATION  *
2970 REM *D'UNE PAGE    *
2980 REM *D'ECRAN       *
2990 REM ****************
3000 IF K=131 THEN 3060
3010 IF K<>146 THEN 3380
3020 PL=3
3030 PC=3
3040 CL=23
3050 CC=30
3060 CALL CLEAR
3070 OPEN #1:"CS1",INTERNAL,INPUT ,FIXED
192
3080 INPUT #1:IP
3090 PRINT "PAGE A RECUPERER-> ";IP
3100 PRINT "CONFIRMATION:O/N?"
3110 CALL KEY(0,K,S)
3120 IF S=0 THEN 3110
3130 IF K<>79 THEN 3270
3140 PAG=IP
3150 FOR I3=1 TO 19 STEP 6
3160 INPUT #1:E$
3170 I4=1
3180 FOR I=I3 TO I3+5
3190 IF I>21 THEN 3270
3200 FOR J=1 TO 28
3210 IF (I>=PL-2)*(J>=PC-2)*(I<=CL-2)*(J<
=CC-2)THEN 3220 ELSE 3230
3220 T$(I,J)=SEG$(E$,I4,1)
3230 I4=I4+1
3240 NEXT J
3250 NEXT I
3260 NEXT I3
3270 CLOSE #1
3280 FOR I=1 TO 21
3290 FOR J=1 TO 28
3300 CALL HCHAR(I+2,J+2,ASC(T$(I,J)))
3310 NEXT J
3320 NEXT I
3330 GOTO 330
3340 REM ***************
3350 REM *NUMEROTATION *
3360 REM *DE PAGE      *
3370 REM ***************
3380 IF K<>142 THEN 640
3390 R$=""
3400 CALL HCHAR(2,18,32,2)
3410 FOR I=18 TO 19
3420 CALL HCHAR(2,I,129)
3430 CALL KEY(0,K,S)
3440 IF S<>1 THEN 3430
3450 IF (K<48)+(K>57)THEN 3430
3460 R$=R$&CHR$(K)
3470 CALL HCHAR(2,I,K)
3480 NEXT I
3490 PAG=VAL(R$)
3500 GOTO 580
3510 REM ***************
3520 REM *ROUTINE      *
3530 REM *D'AFFICHAGE  *
3540 REM ***************
3550 FOR I=1 TO LEN(L$)
```

```
3560 CALL HCHAR(LA,CA+I-1,ASC(SEG$(L$,I,1
)))
3570 NEXT I
3580 RETURN
3590 REM ***************
3600 REM *ROUTINE DES   *
3610 REM *MESSAGES      *
3620 REM ***************
3630 CALL HCHAR(24,1,133,32)
3640 LA=24
3650 CA=3
3660 GOSUB 3550
3670 RETURN
```

Caractères ASCII dans la page graphique

Jean-Luc Bazanegue

En utilisant les routines graphiques publiées dans les numéros 4 et 5 de "99 Magazine", nous nous sommes rendu compte d'une importante lacune du système : il n'était pas possible d'afficher simplement du texte dans la page graphique. Nous nous devions donc de remédier à cet état de fait, en vous proposant une nouvelle routine en assembleur, baptisée *LETTRE* et propre à effectuer ce travail.

Utilisation de la routine

La méthode est la même que pour les routines précédentes, à savoir :

passage des paramètres requis par le programme en assembleur au moyen de *CALL LOAD;*
appel de la routine par le sous-programme *CALL LINK.*

Le passage des paramètres se fait de la manière suivante :

CALL LOAD (−24564, TYPEL, CODEL, LIGNEL, COLONL, CONL, COFFL)

TYPEL, pour *TYPE de Lettre*, détermine la police de caractères utilisée. Si l'argument est égal à zéro, le jeu de caractère "classique" sera employé; avec un argument différent de zéro, la routine récupérera les descripteurs utilisés pour les caractères de la mire d'introduction (les lettres sont dessinées dans une matrice 8⋆6 au lieu de 7⋆5).

Si votre programme Basic est important, il faudra utiliser les caractères de la mire avec prudence; en effet, la table des descripteurs employée pour cette police particulière n'étant pas protégée, elle peut très bien être "écrasée" par un programme envahissant. Dans le cas où cette mésaventure vous arriverait, cela ne "planterait" pas le programme; vous pourriez tout au plus obtenir d'étranges caractères.

CODEL, pour *CODE de la Lettre*, autorise la transmission du code de caractère. Les arguments valides varient en fonction de *TYPEL* :

si *TYPEL* est égal à 0, *CODEL* doit être supérieur à 32 et inférieur à 160. Le fait de pouvoir passer des arguments supérieurs à 127 autorise l'affichage de caractères préalablement définis par l'instruction *CALL CHAR;*

si *TYPEL* est différent de zéro, l'argument doit être supérieur à 32 et inférieur à 91.

LIGNEL et **COLONL** indiquent, bien sûr, où sera affiché le caractère. Les coordonnées sont les mêmes qu'en Basic (1 à 24 pour *LIGNEL* et 1 à 32 pour *COLONL*).

CONL et **COFFL** déterminent les couleurs des points "on" et "off" du caractère concerné. Puisque nous appelons une routine écrite en assembleur, l'argument doit êre le code de la couleur en Basic, moins un (0 à 15).

La routine est protégée contre les arguments invalides qui pourraient perturber son fonctionnement; si elle rencontre une valeur incompatible avec les indications fournies ci-dessus, rien ne se passe et la main est redonnée au programme Basic.

Concrètement, pour afficher un "A" (8 ⋆ 6) à la ligne 5, colonne 10, et en noir sur blanc, il faut utiliser :

CALL LOAD (−24564, 1, 65, 5, 10, 1, 15)
CALL LINK ("LETTRE")

Lorsque vous transmettez des arguments à la routine *LETTRE*, ils sont pris par défaut jusqu'à ce que d'autres valeurs soient spécifiées; il n'est donc pas toujours nécessaire de communiquer tous les arguments. Par exemple, si, après avoir choisi un type de lettre, vous souhaitez conserver le même, le premier argument peut être le code du caractère. Dans ce cas, il faut changer l'adresse du premier "poke" qui devient −24563. De même, si vous ne désirez pas changer les couleurs, il est inutile de les transmettre à nouveau.

Pour afficher un mot ou une phrase dans la page graphique, un petit sous-programme, semblable à ceux que l'on utilise en Basic TI pour simuler le *DISPLAY AT* du Basic Etendu, peut s'avérer très utile. Ce qui suit pourrait faire l'affaire (ne pas oublier de charger les routines graphiques !).

```
100 T$ = "Organigramme"
110 TYPEL = 0
120 LIGNEL = 3
130 COLONL = 5
140 CONL = 7
150 COFFL = 12
160 GOSUB 1000

1000 PC = 1
1010 FOR I = COLONL TO COLONL + LEN(T$) − 1
1020 CALL LOAD (−24564, TYPEL, ASC (SEG$ (T$, PC, 1)), LIGNEL, I, CONL, COFFL)1030 CALL LINK ( "LETTRE" )
1040 PC = PC +1
1050 NEXT I
1060 CALL LINK ( "IMAGE" )
1070 RETURN
```

Avant de passer aux explications concernant le fonctionnement de la routine en assembleur, nous devons signaler à nos lecteurs ayant fait l'acquisition de la disquette *"Routines graphiques"* qu'ils sont déjà en possession de ce petit supplément.

Fonctionnement de la routine

Comme pour la routine *LIGNE* du précédent numéro, seul le listing de la nouvelle routine apparaît dans ces pages. Pour avoir l'intégralité du programme, vous pouvez vous reporter aux numéro 4 et 5 de "99 Magazine". Cette fois, nous avons volontairement laissé les numéros de lignes issus de l'assemblage, afin de faciliter les explications qui vont suivre.

0279. Il s'agit du point d'entrée dans la routine. *MOVB @ CODEL,R14* déplace une copie du contenu de l'octet situé à l'adresse >A00D (−24563) vers l'octet de poids fort du registre 14 (voir l'article "Les modes d'adressage en Assembleur", publié dans le numéro 3). A ce stade, le registre 14 contient donc le code du caractère que l'on désire afficher plus, dans l'octet de poids faible, une valeur indéterminée.

0280. L'instruction *SRL* exécute un décalage logique vers la droite. Dans le cas qui nous intéresse, il s'agit de décaler le registre 14 de huit positions pour que le code de notre caractère soit dans l'octet de poids faible, et aussi de nous débarrasser de ce qui encombre l'octet de poids faible. Si l'on admet que le code transmis est 65 nous avons, dans le registre 14 :

01000001 XXXXXXXX (les "X" représentent la valeur indéterminée dont nous parlions précédemment).

Après l'exécution de la ligne 0280, le registre 14 contient :

00000000 01000001.

En effet, l'instruction SRL met à 0 tous les bits qui prennent la place des bits décalés, contrairement à l'instruction *SRA* (décalage arithmétique vers la droite) qui remplace tous les bits décalés par le bit de signe. Nous pouvons noter que pour des valeurs inférieures à 128 (dans l'octet de poids fort de R14), *SRA R14,8* et *SRL R14,8* auraient donné le même résultat puisque, dans ce cas, le bit de signe est à zéro. Les bits décalés vers la droite en dehors du registre sont perdus, sauf le dernier bit éjecté pendant l'opération car il "tombe" dans le "carry" (retenue) du registre d'état.

0281. Le microprocesseur réalise ici une opération très simple : la comparaison immédiate du registre 14 avec la valeur 32. En fonction du résultat de cette comparaison, les bits du registre d'état "égal", "logiquement supérieur" et "arithmétiquement supérieur" sont mis à 1 ou à 0 (1 = vrai, 0 = faux).

0282. L'instruction *JH* (saut si logiquement supérieur) nous permet d'utiliser le résultat de la comparaison précédente et ainsi intervenir sur le déroulement du programme. Nous trouvons ici une des différences fondamentales entre le Basic et l'Assembleur :
en Basic, le test et le branchement en fonction du résultat sont effectués simultanément (IF X = Y THEN 100, par exemple); en Assembleur, un test permettant un détournement utilise toujours le résultat de la dernière opération effectuée par le microprocesseur.

Dans notre programme, si le bit d'état "logiquement supérieur" est à 1 et si le bit "égal" est à zéro, le programme est dirigé vers l'étiquette *L0* puisque le code du caractère est supérieur à 32. Dans le cas contraire, le contrôle est passé à la ligne suivante.

0283. Si le code du caractère proposé est inférieur à 33, nous avons affaire à une valeur qui pourrait perturber le fonctionnement de la routine; le traitement doit donc être interrompu. Le but de la ligne 0283 est de diriger le programme vers la routine baptisée *RETOUR* (voir numéro 4) qui redonne la main au programme Basic.

0284. Nous réalisons ici une opération semblable à celle de la ligne 0279 mais le but est différent. Lorsque l'instruction *MOVB* (ou *MOV* pour un mot de seize bits) est employée, la valeur déplacée est comparée à zéro et le contenu du registre d'état est modifié en conséquence (mêmes bits que pour l'instruction CI). Ceci va nous permettre de déterminer le type de caractère choisi.

0285. L'instruction *JEQ* (saut si égal) provoque un branchement à l'adresse symbolisée par l'étiquette L1 si le bit d'état "égal" est à 1, donc si le contenu de l'adresse *TYPEL* contenait une valeur nulle. Dans ce cas, nous savons que le programme Basic demande un caractère standard. Si le résultat du test est négatif, nous passons à la ligne suivante.

0286. L'adresse 2304, en mémoire vive vidéo, contient le premier des huit octets définissant le caractère "espace" inscrit dans une matrice 8 ⋆ 7. Nous stockons cette adresse dans le registre 13 pour une utilisation ultérieure.

0287. Cette ligne, comme la précédente, est utilisée uniquement dans le cas où le programme Basic demande un caractère 8 ⋆ 7. Dans ce mode, la définition des caractères s'arrête à la lettre "Z" (code 90); nous devons donc comparer le contenu du registre 14 (qui contient le code du caractère demandé) avec la valeur 91.

0288. Si la valeur est inférieure à 91, les bits d'état "égal" et "logiquement supérieur" sont à 0 et l'instruction *JL* (saut si logiquement inférieur) dirige le programme vers l'étiquette "L2". L'instruction *JL* ne peut pas être remplacée par *JLT* (saut si arithmétiquement inférieur) car 128, par exemple, devient inférieur à 91 puisque le bit de signe est à 1.

0289. Si le registre 14 contient une valeur supérieure à 90, la routine redonne la main au Basic. L'instruction JMP provoque un saut inconditionnel à l'adresse indiquée explicitement, ou sous une forme symbolique.

0290. La routine exécute cette ligne uniquement si un caractère standard est requis; le type d'opération réalisé est semblable à celui de la ligne 0286, mais nous plaçons cette fois dans le registre 13 l'adresse 1024, qui correspond au premier des huit octets définissant le caractère "espace" en 7⋆5 points.

0291. Nous faisons une comparaison entre le registre 14 et la valeur 159 qui est le dernier code possible à partir du Basic. Le test, contrairement à celui de la ligne 0286, est effectué sur la valeur maximale admise au lieu de la valeur suivant la valeur maximale; ceci va nous permettre de gagner une instruction.

0292. Puisque nous avons comparé le registre 14 avec la valeur 159, nous pouvons utiliser l'instruction *JH* (saut si logiquement supérieur) pour redonner la main au Basic si le code transmis est incorrect. Si nous avions utilisé 160 pour le test, nous aurions dû employer :

JL L2
JMP L3

au lieu de :

JH L3.

0293. Nous retrouvons ici un segment commun aux deux jeux de caractères. L'instruction *AI* (Addition Immédiate) ajoute au registre concerné la valeur qui se trouve en mémoire immédiatement après l'instruction. Dans notre routine, cette valeur est négative; cela revient à faire une "soustraction immédiate". Après l'exécution de cette instruction, le registre 14 (dans lequel se trouve toujours le code de notre caractère)

contient le code moins 32, soit 33 pour un "A", 34 pour un "B", etc...

0294. Nous avons, dans le registre 13, l'adresse du premier des huit octets définissant le caractère 32, qui correspond maintenant à la valeur 0; il faut maintenant calculer l'adresse du premier octet définissant le caractère qui nous intéresse. Pour cela, nous utiliserons la formule :

adresse de base + (code du caractère − 32) ⋆ 8.

L'adresse de base se trouve dans le registre 13 et le code du caractère moins 32 est dans le registre 14. Pour la multiplication par huit, nous utilisons l'instruction *SLA* (décalage arithmétique de n position vers la gauche). En effet, chaque fois qu'un registre est décalé d'une position vers la gauche, son contenu est multiplié par deux à condition que la valeur de départ soit inférieure à 32767. Ainsi, en décalant le registre de 3 positions, son contenu est multiplié par 8 (2⋆2⋆2 ou 2̂3). Il est possible de faire des divisions par un multiple de 2 en utilisant l'instruction *SRL*.

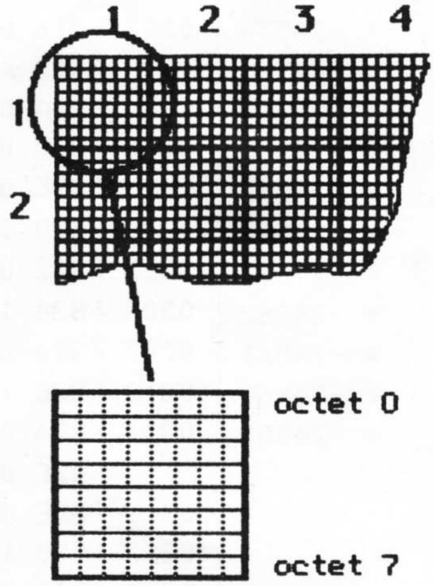
0295. Le registre 14 contient maintenant le décalage par rapport à l'adresse de base. L'instruction *A* (Addition) ajoute cette valeur au registre 13, qui devient l'adresse réelle du premier octet dont nous aurons besoin.

0296 à **0299**. Le numéro de colonne poké à l'adresse −24561 est reporté dans l'octet de poids fort du registre 14. Le traitement est exactement le même que celui utilisé aux lignes 0279 à 0282 pour le code de la lettre. Si le numéro de colonne est supérieur à 32, il y a retour au Basic.

0300. Le registre 14 est décrémenté de 1 de façon à obtenir une valeur comprise entre 0 et 31 inclus.

0301. Lorsque l'on utilise l'instruction *DEC*, le bit d'état "carry" est toujours mis à 1 sauf lorsque l'on passe de 0 à −1. Si le programme Basic transmet zéro comme numéro de ligne, le fait de décrémenter le registre 14 va faire passer son contenu de 0 à −1, donc le bit "carry" sera à 0. Dans ce cas, l'instruction *JNC* (saut si pas de retenue) oriente la routine vers le programme Basic.

0302. La multiplication par huit nous fournit le décalage par rapport à l'adresse de base du tampon "TIMAGE", et cela en supposant que la ligne 1 soit choisie. Je m'explique; les octets constituant la table d'écran en mode Bit Map sont disposés de cette façon :



Donc, pour la colonne 3 (qui devient la colonne 2 après l'exécution de l'instruction *DEC R14*) nous obtenons dans le registre 14 : 2 ⋆ 8 soit 16 ou >10 en hexadécimal, ce qui correspond à la valeur qu'il faudra ajouter à l'adresse de base du tampon TIMAGE pour la ligne 1.

0303 à **0308**. Traitement en tous points identique à celui du numéro de colonne, mis à part le premier test de validité, qui est effectué par rapport à 24 au lieu de 32.

0309. L'instruction *MOV R12,R12* met le bit d'état "egal" à 1 si la valeur contenue par R12 est nulle.

0310. Si le bit d'état est à 1, nous savons que le programme Basic a demandé l'affichage d'un caractère sur la première ligne (le registre 12 est décrémenté de 1 à la ligne 0307; ainsi, pour la ligne 1, le registre 12 contient maintenant 0). Dans ce cas, l'adresse à modifier dans le tampon TIMAGE se trouve déjà dans le registre 14 et nous pouvons passer les trois lignes suivantes, réservées aux numéros de lignes supérieurs à 1.

0311 à **0313**. En mode Bit Map, chaque ligne sur l'écran utilise 8 ⋆ 32 = 256 octets. Si nous ajoutons 256 multiplié par le numéro de ligne moins 1 (registre 12) au registre 14, nous obtenons l'adresse du premier des huit octets à modifier dans le tampon TIMAGE. Au lieu de faire une multiplication, nous ajoutons 256 au registre 14, le registre 12 est ensuite décrémenté et l'opération est recommencée jusqu'à ce que le registre 12 soit égal à zéro.

0314. A partir de cette ligne, nous nous occupons des couleurs de notre caractère. Ici, il n'y a pas de contrôle de validité puisque nous ne conserverons que les quatre bits de poids faible des valeurs pokées.

Après l'exécution de la ligne 0314, la couleur des points "off" est située dans l'octet de poids fort du registre 9. Pour la couleur 15 (blanc), R9 est égal à 00001111 00000000.

0315. Mise à zéro de tous les bits du registre 10.

0316. La couleur des points "on" est reportée dans l'octet de poids fort du registre 10. Pour la couleur 1 (noir), R9 est égal à 00000001 00000000.

0317. Le contenu du registre 10 est décalé de 12 positions vers la droite mais il s'agit d'un décalage circulaire. Les bits qui sortent du registre ne sont pas perdus puisqu'ils sont reportés dans le bit de poids fort. Le registre 9 contient maintenant 00010000 00000000.

0318. *SOC R10,R9* met à 1 les bits de R9 dont les positions correspondent aux bits à 1 du registre 10.

Avant exécution :

R10 = 00001111 00000000
R9 = 00010000 00000000.

Après exécution :

R9 = 00011111 00000000.

La couleur des points "on" est dans le quartet de gauche de l'octet de poids fort du registre 9 (0001 = 1), alors que la couleur des points "off" se trouve dans le quartet de droite du même octet (1111 = 15).

0319. Chargement de l'adresse de base du tampon TCOUL dans le registre 0.

0320. Puisque la table des couleurs est structurée de la même manière que la table d'écran, nous ajoutons au registre 0 la même valeur de décalage que pour cette dernière, et nous obtenons l'adresse du premier des huit octets à modifier dans le tampon TCOUL.

Le reste de la routine consiste en la modification des tampons TCOUL et TIMAGE en fonction des paramètres établis précédemment. La méthode employée étant la même que pour les routines du numéro 4 de "99", je pense que nous pouvons interrompre ces explications relativement longues pour une routine exécutée en quelques millièmes de secondes ! ■

```
0001                DEF  EFFT,EFFP,EFFC
0002                DEF  AXES,POINT,IMAGE
0003                DEF  LIGNE,LETTRE
0004
```

```
0005 A000                AORG >A000
0006 A000 0000 COORD   DATA 0       * -24576
0007 A002 0000 COULP   DATA 0       * -24574
0008 A004 0000 COULE   DATA 0       * -24572
0009 A006 0000 POSAXE  DATA 0       * -24570
0010 A008 0000 LIGNED  DATA 0       * -24568
0011 A00A 0000 LIGNEA  DATA 0       * -24566
0012 A00C   00 TYPEL   BYTE 0       * -24564
0013 A00D   00 CODEL   BYTE 0       * -24563
0014 A00E   00 LIGNEL  BYTE 0       * -24562
0015 A00F   00 COLONL  BYTE 0       * -24561
0016 A010   00 CONL    BYTE 0       * -24560
0017 A011   00 COFFL   BYTE 0       * -24559
0018 A012 0000 REG11   DATA 0
0019 A014 0000 CR7     DATA 0
0020 A016 0000 CR8     DATA 0
0278
0279 FAD0 D3A0 LETTRE  MOVB @CODEL,R14
     FAD2 A00D
0280 FAD4 098E         SRL  R14,8
0281 FAD6 028E         CI   R14,32
     FAD8 0020
0282 FADA 1B02         JH   L0
0283 FADC 0460 L3      B    @RETOUR
     FADE FAC8
0284 FAE0 D3E0 L0      MOVB @TYPEL,R15
     FAE2 A00C
0285 FAE4 1306         JEQ  L1
0286 FAE6 020D         LI   R13,2304
     FAE8 0900
0287 FAEA 028E         CI   R14,91
     FAEC 005B
0288 FAEE 1A06         JL   L2
0289 FAF0 10F5         JMP  L3
0290 FAF2 020D L1      LI   R13,1024
     FAF4 0400
0291 FAF6 028E         CI   R14,159
     FAF8 009F
0292 FAFA 1BF0         JH   L3
0293 FAFC 022E L2      AI   R14,-32
     FAFE FFE0
0294 FB00 0A3E         SLA  R14,3
0295 FB02 A34E         A    R14,R13
0296 FB04 D3A0         MOVB @COLONL,R14
     FB06 A00F
0297 FB08 098E         SRL  R14,8
0298 FB0A 028E         CI   R14,32
     FB0C 0020
0299 FB0E 1BE6         JH   L3
0300 FB10 060E         DEC  R14
0301 FB12 17E4         JNC  L3
0302 FB14 0A3E         SLA  R14,3
0303 FB16 D320         MOVB @LIGNEL,R12
     FB18 A00E
0304 FB1A 098C         SRL  R12,8
0305 FB1C 028C         CI   R12,24
     FB1E 0018
0306 FB20 1BDD         JH   L3
0307 FB22 060C         DEC  R12
0308 FB24 17DB         JNC  L3
0309 FB26 C30C         MOV  R12,R12
0310 FB28 1304         JEQ  L4
0311 FB2A 022E L5      AI   R14,256
     FB2C 0100
0312 FB2E 060C         DEC  R12
0313 FB30 16FC         JNE  L5
0314 FB32 D260 L4      MOVB @COFFL,R9
     FB34 A011
0315 FB36 04CA         CLR  R10
0316 FB38 D2A0         MOVB @CONL,R10
     FB3A A010
0317 FB3C 0BCA         SRC  R10,12
0318 FB3E E24A         SOC  R10,R9
0319 FB40 0200         LI   R0,TCOUL
     FB42 2676
0320 FB44 A00E         A    R14,R0
0321 FB46 0201         LI   R1,8
     FB48 0008
0322 FB4A DC09 L6      MOVB R9,*R0+
0323 FB4C 0601         DEC  R1
0324 FB4E 16FD         JNE  L6
0325 FB50 0200         LI   R0,TIMAGE
     FB52 E018
0326 FB54 A00E         A    R14,R0
0327 FB56 0201         LI   R1,8
     FB58 0008
0328 FB5A 06CD         SWPB R13
0329 FB5C D80D         MOVB R13,@>8C02
     FB5E 8C02
0330 FB60 06CD         SWPB R13
0331 FB62 D80D         MOVB R13,@>8C02
     FB64 8C02
0332 FB66 1000         NOP
0333 FB68 DC20 L7      MOVB @>8800,*R0+
     FB6A 8800
0334 FB6C 0601         DEC  R1
0335 FB6E 16FC         JNE  L7
0336 FB70 10B5         JMP  L3
0337
0338                   END
```

Musik

Michel Legault

Ce programme, écrit en **Basic TI,** n'a pas la prétention de remplacer un instrument de musique; son but est essentiellement éducatif et concerne l'apprentissage du solfège. Cependant, il pourra quand même servir à reconnaître un air à partir d'une feuille de musique.

Le clavier est organisé sur le même principe qu'un piano, avec ses touches blanches et noires.
Tous les caractères alphanumériques et les chiffres *0* à *9* sont susceptibles de représenter une note.

La première et la troisième ligne du clavier représentent exclusivement des dièses et des bémols, alors que la deuxième et la quatrième ligne sont liées aux notes régulières. La touche *Q* correspond au *SI* le plus grave, et les notes se termine avec la touche *M* qui représente le *RE* le plus aigu.

Les fonctions

/ (barre oblique)

Changement alternatif du mode note au mode accord. Lorsque vous pressez cette touche, le programme se met dans le mode opposé à celui qui est affiché. Vous êtes en mode accord quand apparaît, à la suite de *ACCORD=*, le caractère 1, et vous êtes en mode note quand apparaît un 0.

En mode note, vous pouvez jouer des notes simples qui apparaîtront sur votre écran les unes après les autres.

En mode accord, vous devez presser trois touches consécutives; chacune des notes s'affichera sur la même colonne et, lors de la pression sur la troisième touche, vous entendrez l'accord. Durant l'entrée des trois notes, il est impossible d'exécuter d'autres fonctions.

CTRL-T

Changement du temps de base. Pressez *CTRL-T* et le temps de base sera affiché, puis entrez un chiffre de *1* à *9* pour donner le nouveau temps. Le calcul du temps de base est effectué de la manière suivante :

temps (en millisecondes) = chiffre ⋆ 100.

. (point)

Changement alternatif entre le mode note pointée et note standard. Lorsque vous pressez cette touche, le programme se met dans le mode opposé à celui qui est affiché. Vous êtes en mode note pointée quand apparaît, à la suite de *POINTEE=*, le caractère . (point), alors que vous êtes en mode note standard dans le cas contraire.

CTRL-A, B, C, D et E

Les touches de contrôle *A* à *E* déterminent les types de notes, ainsi que le temps relatif au temps de base :

- A = double croche = 1/4 de temps;
- B = croche = 1/2 temps;
- C = noire = 1 temps;
- D = blanche = 2 temps;
- E = ronde = 4 temps.

Sélectionnez la note désirée et elle sera affichée à la suite de *NOTE=*. Ceci sera utilisé pour les prochaines notes jouées.

CTRL-S

Vous pouvez insérer un silence entre les notes. Pressez *CTRL-S* et un silence d'un temps sera enregistré, avec l'affichage d'un symbole sur la troisième ligne de la portée.

SHIFT – et +

Permet de contrôler le volume :

- \+ augmente le volume pour les notes suivantes;
- – diminue le volume pour les notes suivantes.

Le volume est affiché à la suite de *VOLUME=* et est représenté par un nombre de *1* à *30*. Plus le nombre est grand et plus le volume est faible.

ENTER

Lorsque vous avez terminé une portée, appuyez sur *ENTER*; le programme jouera la musique enregistrée jusque là et initialisera une nouvelle portée.

CTRL-M

Permet de jouer la musique de la portée courante (seulement les notes affichées sur la portée).

CTRL-S

Efface la portée courante.

FCTN-B

Reprise de l'enregistrement musical depuis le début. ■

```
100 REM **********************
110 REM *        MUSIK       *
120 REM **********************
130 REM *       BASIC TI     *
140 REM **********************
150 REM
160 CALL CLEAR
170 CALL COLOR(14,5,16)
180 CALL COLOR(15,5,16)
190 CALL COLOR(16,5,16)
200 CALL COLOR(13,2,15)
210 CALL COLOR(5,2,12)
220 CALL COLOR(6,2,12)
230 CALL COLOR(7,2,12)
240 DIM L(43),F(43),D(5),E(5,100)
250 FOR I=1 TO 43
260 READ F(I),L(I)
270 NEXT I
280 DATA 554,-13,233,-22,0,0,277,-20,311,
-19,0,0,370,-17,416,-16,466,-15,0,0,0,0
,0,0
290 DATA 0,0,0,0,0,0,0,0,0,0,622,-12,988,
7,784,9,740,-10,294,19,831,-9,932,-8,0,
0
300 DATA 494,14,1109,-6,1245,-5,0,0,1175,
5,1047,6,523,13,587,12,247,21,330,18,0,
0
310 DATA 349,17,440,15,880,8,262,20,698,1
```

```
0,392,16,659,11
320 READ D(1),D(2),D(3),D(4),D(5)
330 DATA .25,.5,1,2,4
340 CALL CHAR(128,"000000FFFF000000")
350 CALL CHAR(129,"07040704347C7830")
360 CALL CHAR(130,"07040404347C7830")
370 CALL CHAR(131,"04040404347C7830")
380 CALL CHAR(132,"04040404344C4830")
390 CALL CHAR(133,"0018244242241800")
400 CALL CHAR(134,"000000FF4242427E")
410 CALL CHAR(136,"0000000000000000")
420 CALL CHAR(137,"3844828282828284")
430 CALL CHAR(138,"000000FFFF000000")
440 CALL CHAR(139,"848888FFFFA0C0C0")
450 CALL CHAR(140,"0000000101020204")
460 CALL CHAR(141,"8080808080808080")
470 CALL CHAR(142,"040808FFFF202040")
480 CALL CHAR(143,"808080FFFF808080")
490 CALL CHAR(144,"4080800000000000")
500 CALL CHAR(145,"6040408080808080")
510 CALL CHAR(146,"80808080808080F8")
520 CALL CHAR(147,"808080FFFF404040")
530 CALL CHAR(148,"030C10FFFF10180C")
540 CALL CHAR(149,"8E8380FFFF988898")
550 CALL CHAR(150,"0080E0FFFF380C04")
560 CALL CHAR(151,"4020202010101008")
570 CALL CHAR(152,"0603000000000000")
580 CALL CHAR(153,"B0E0808080808080")
590 CALL CHAR(154,"0404040404040808")
600 CALL CHAR(155,"080C06FFFF000000")
610 CALL CHAR(156,"000000FFFFC0701F")
620 CALL CHAR(157,"808080FFFF8386FC")
630 CALL CHAR(158,"183060FFFF000000")
640 CALL CHAR(159,"8181424224241800")
650 FOR I=9 TO 20
660 FOR J=2 TO 5
670 READ CS
680 CALL HCHAR(I,J,CS)
690 NEXT J
700 NEXT I
710 DATA 136,136,137,136,138,138,139,138,
136,140,141,136,138,142,143,138
720 DATA 140,144,141,136,142,138,143,138,
145,136,146,136,147,148,149,150,151,152
,153,154
730 DATA 155,156,157,158,136,136,141,136,
136,159,136,136
740 DIM S$(2)
750 S$(1)="ACCORD =0  TEMPS DE BASE=3"
760 S$(2)="POINTEE=   NOTE="&CHR$(131)&"
VOLUME=3"
770 FOR J=1 TO 2
780 FOR I=1 TO LEN(S$(J))
790 CALL HCHAR(J,I+2,ASC(SEG$(S$(J),I,I))
)
800 NEXT I
810 NEXT J
820 PTE=1
830 CH=131
840 TEM=300
850 VOL=3
860 CUR=0
870 GOSUB 1890
880 C=7
890 CALL KEY(0,K,S)
900 IF S<>1 THEN 890
910 IF K<>61 THEN 930
920 END
930 IF K=13 THEN 870
940 IF K=6 THEN 860
950 IF (K<48)+(K>90)THEN 1270
960 IF F(K-47)=0 THEN 890
970 IF C>29 THEN 890
980 IF ACC=0 THEN 1030
990 IF CUR>99 THEN 1220
1000 GOSUB 2150
1010 GOSUB 2010
1020 GOTO 890
1030 CALL HCHAR(ABS(L(K-47)),C,CH)
1040 C=C+1
1050 IF L(K-47)>0 THEN 1080
1060 CALL HCHAR(ABS(L(K-47)),C,35)
1070 C=C+1
1080 IF PTE=1 THEN 1130
1090 CALL HCHAR(ABS(L(K-47)),C,46)
1100 C=C+1
1110 PTE=1
1120 CALL HCHAR(2,11,32)
1130 CALL SOUND(D(CH-128)*TEM*PTE,F(K-47)
,VOL)
1140 CUR=CUR+1
1150 IF CUR>100 THEN 1220
1160 E(1,CUR)=D(CH-128)*TEM*PTE
1170 E(2,CUR)=F(K-47)
1180 E(3,CUR)=VOL
1190 E(4,CUR)=E(2,CUR)
1200 E(5,CUR)=E(2,CUR)
1210 GOTO 890
1220 M$="MEMOIRE DES NOTES REMPLIE"
1230 FOR I=1 TO LEN(M$)
1240 CALL HCHAR(24,I+2,ASC(SEG$(M$,I,I)))
1250 NEXT I
```

```
1260 GOTO 890
1270 IF (K>128)*(K<134)THEN 1280 ELSE 131
0
1280 CH=K
1290 CALL HCHAR(2,19,CH)
1300 GOTO 890
1310 IF K<>46 THEN 1390
1320 IF PTE=1 THEN 1360
1330 PTE=1
1340 CALL HCHAR(2,11,32)
1350 GOTO 890
1360 PTE=1.5
1370 CALL HCHAR(2,11,46)
1380 GOTO 890
1390 IF K<>43 THEN 1430
1400 IF VOL>29 THEN 890
1410 VOL=VOL+1
1420 GOTO 1470
1430 IF K<>45 THEN 1510
1440 IF VOL<2 THEN 890
1450 VOL=VOL-1
1460 CALL HCHAR(2,28,32,2)
1470 FOR I=1 TO LEN(STR$(VOL))
1480 CALL HCHAR(2,I+27,ASC(SEG$(STR$(VOL)
,I,I)))
1490 NEXT I
1500 GOTO 890
1510 IF K<>47 THEN 1580
1520 IF ACC=0 THEN 1550
1530 ACC=0
1540 GOTO 1560
1550 ACC=1
1560 CALL HCHAR(1,11,ASC(STR$(ACC)))
1570 GOTO 890
1580 IF K<>147 THEN 1700
1590 IF C>29 THEN 890
1600 CALL HCHAR(16,C,134)
1610 CUR=CUR+1
1620 IF CUR>100 THEN 32767
1630 E(1,CUR)=TEM
1640 E(2,CUR)=44733
1650 E(3,CUR)=30
1660 E(4,CUR)=44733
1670 E(5,CUR)=44733
1680 C=C+1
1690 GOTO 890
1700 IF K<>141 THEN 1760
1710 IF (CUR=0)+(C=7)THEN 890
1720 FOR I=DP+1 TO CUR
1730 CALL SOUND(E(1,I),E(2,I),E(3,I),E(4,
I),E(3,I),E(5,I),E(3,I))
1740 NEXT I
1750 GOTO 890
1760 IF K<>8 THEN 1810
1770 CUR=DP
1780 C=7
1790 GOSUB 1930
1800 GOTO 890
1810 IF K<>148 THEN 890
1820 CALL HCHAR(1,28,32)
1830 CALL KEY(0,K,S)
1840 IF S<>1 THEN 1830
1850 IF (K<49)+(K>57)THEN 1830
1860 TEM=(K-48)*100
1870 CALL HCHAR(1,28,K)
1880 GOTO 890
1890 IF CUR=0 THEN 1930
1900 FOR I=1 TO CUR
1910 CALL SOUND(E(1,I),E(2,I),E(3,I),E(4,
I),E(3,I),E(5,I),E(3,I))
1920 NEXT I
1930 FOR J=3 TO 22
1940 CALL HCHAR(J,6,32,26)
1950 NEXT J
1960 FOR J=10 TO 18 STEP 2
1970 CALL HCHAR(J,6,128,26)
1980 NEXT J
1990 DP=CUR
2000 RETURN
2010 CUR=CUR+1
2020 E(1,CUR)=D(CH-128)*TEM*PTE
2030 E(2,CUR)=F(K-47)
2040 GOSUB 2120
2050 E(4,CUR)=F(K-47)
2060 GOSUB 2120
2070 E(5,CUR)=F(K-47)
2080 E(3,CUR)=VOL
2090 CALL SOUND(E(1,CUR),E(2,CUR),E(3,CUR
),E(4,CUR),E(3,CUR),E(5,CUR),E(3,CUR))
2100 C=C+3
2110 RETURN
2120 CALL KEY(0,K,S)
2130 IF (K<48)+(K>90)+(S<>1)THEN 2120
2140 IF F(K-47)=0 THEN 2120
2150 CALL HCHAR(ABS(L(K-47)),C,CH)
2160 IF L(K-47))0 THEN 2180
2170 CALL HCHAR(ABS(L(K-47)),C+1,35)
2180 IF PTE=1 THEN 2220
2190 PTE=1
2200 CALL HCHAR(2,11,32)
2210 CALL HCHAR(ABS(L(K-47)),C+2,46)
2220 RETURN
```

Copie d'écran sur Seikosha GP100

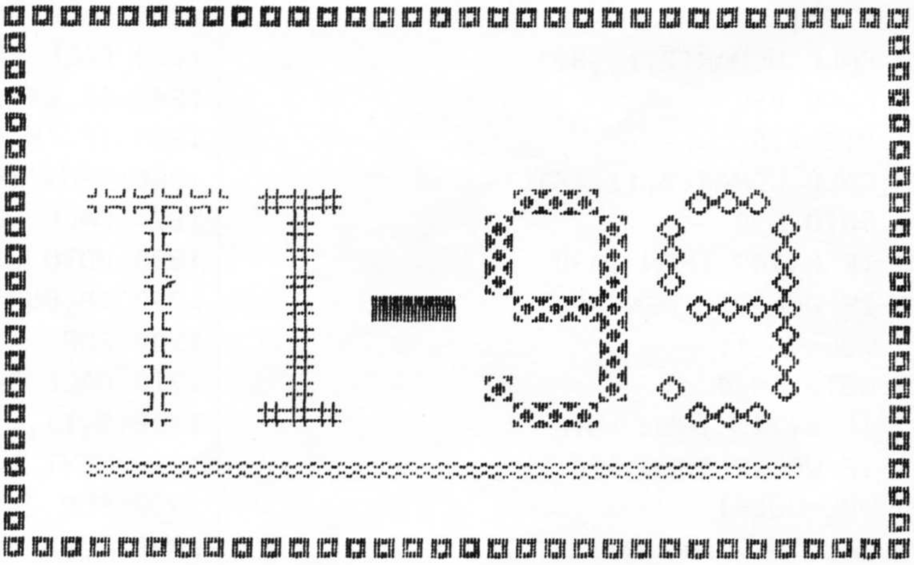
Mansur Tufail

Ce programme écrit en **Basic Etendu** permet de faire une copie d'écran avec une résolution légèrement inférieure à celle du TI (255 ⋆ 192 au lieu de 256 ⋆ 192).

La Seikosha GP100 utilise une matrice à 7 points, alors que la définition des caractères du TI est réalisée dans une matrice 8 ⋆ 8. Cela n'est pas sans poser quelques problèmes lorsque l'on veut faire une copie d'écran.

Dans ce programme, les descripteurs hexadécimaux non traités dans une ligne (huitième point) sont concaténés avec ceux de la ligne suivante, ce qui nécessite la réservation de 32 variables (P$(32)). Lorsque ces 32 variables (une par colonne) atteignent une longueur de 14 descripteurs hexadécimaux, on imprime cette ligne "fictive" supplémentaire, sans incrémenter le numéro de ligne (variable Z1).

Pour la même raison, la variable Z1 varie de 1 à 25 (alors qu'il n'y a que 24 lignes), de façon à imprimer ce qui reste dans les variables P$(32) après traitement de la dernière ligne. Le reste du programme correspond à ce que l'on peut trouver dans tous les programmes de "hard copy". ■




```
10000 ! ********************
10010 ! *  Copie  d'ecran   *
10020 ! *  sur imprimante   *
10030 ! *     Seikosha      *
10040 ! ********************
10050 ! *   Basic Etendu    *
10060 ! ********************
10070 !
10080 DIM P$(32),G(8)
10090 OPEN #1:"PIO",OUTPUT,VARIABLE 255
10100 PRINT #1:CHR$(8)
10110 K$="" :: FOR I=1 TO 32 :: P$(I)=""
:: NEXT I
10120 FOR Z1=1 TO 25
10130 Z1$="" :: IMAX=8
10140 FOR Z2=1 TO 32
10150 IF LEN(P$(Z2))=14 OR Z1=25 THEN C$=
"" :: GOTO 10180
10160 CALL GCHAR(Z1,Z2,Z3)
10170 CALL CHARPAT(Z3,C$)
10180 C$=P$(Z2)&C$
10190 IF C$=K$ THEN 10380 ELSE K$=C$
10200 J=0 :: N=1 :: G(1)=128 :: G(2)=128
:: G(3)=128 :: G(4)=128 :: G(5)=128 ::
G(6)=128 :: G(7)=128 :: G(8)=128
10210 FOR I=1 TO 14
10220 S$=SEG$(C$,I,1)
10230 IF S$>"9" THEN 10280
10240 IF S$="" THEN 10360
10250 S=VAL(S$)
10260 IF S=0 THEN 10360
10270 GOTO 10290
10280 S=ASC(S$):: S=S-55
10290 IF S>7 THEN G(1+J)=G(1+J)+N
10300 IF (S>3 AND S<8)OR S>11 THEN G(2+J)
=G(2+J)+N
10310 IF (S=2 OR S=3 OR S=6 OR S=7 OR S=1
0 OR S=11 OR S=14 OR S=15)THEN G(3+J)=G
(3+J)+N
10320 S=S-2
10330 IF SGN(S)=0 THEN 10360
10340 IF SGN(S)=1 THEN 10320
10350 G(4+J)=G(4+J)+N
10360 IF J=0 THEN J=4 ELSE J=0 :: N=N*2
10370 NEXT I
10380 IF Z2=32 THEN IMAX=7
10390 FOR I=1 TO IMAX
10400 Z1$=Z1$&CHR$(G(I))
10410 NEXT I
10420 IF Z1=25 THEN 10440
10430 P$(Z2)=SEG$(C$,15,LEN(C$)-14)
10440 NEXT Z2
10450 PRINT #1:Z1$
10460 IF LEN(P$(32))=14 THEN 10130
10470 NEXT Z1
10480 CLOSE #1
10490 STOP
```

Nouvelles fonctions avec la Mini-Mémoire

Denise Amrouche

Nous vous proposons un programme en Assembleur qui vous fournit de nouvelles fonctions, appelées du Basic, qui s'exécutent très rapidement et vous permettrons d'améliorer la présentation de vos programmes. Vous pouvez bien sûr créer vous-même d'autres fonctions.

Attention : l'accès direct au langage machine fait que certaines erreurs peuvent être fatales (contrôle perdu, programme disparu), ainsi nous vous recommandons de sauvegarder votre programme Basic avant de le tester.

Mode d'emploi

1) Insérez votre Mini-mémoire. Chargez-y le programme à l'aide de *EASY-BUG* :

- commande *L* si vous disposez de la cassette;
- sinon commande M. Vous commencez, par exemple, par M7118, tapez 04 (ENTER), apparaît M7119, tapez EO (ENTER), apparaît M711A, tapez 83 (ENTER), etc...

Sauvegardez sur cassette, commande *S7000 TO 8000*

2) Vous disposez maintenant de toutes les fonctions détaillées ci-après. Le programme ESSAI n'est qu'un exemple simple des possibilités offertes. Après l'avoir étudié de près, l'avoir modifié etc... vous serez prêt à faire vos propres applications.

Note : vous passez d'un exemple au suivant en appuyant sur une touche quelconque.

Notez la vitesse du programme *CHAR* qui vous permet, en redéfinissant 4 caractères d'un coup, de voir le moulin tourner avec réalisme et sans à-coups.

Dans le troisième exemple, vous pouvez redéfinir tout caractère de code ASCII de 1 à 159. Pour le caractère 0, vous détruisez éventuellement le >DO qui est à l'adresse 768 de la mémoire vidéo et qui indique qu'il n'y a pas de sprites définis, vous voyez alors apparaître des sprites.

Ainsi, vous pouvez désormais utiliser en Basic des caractères de 1 à 159 (ou tout au moins essayer car il se peut que certains programmes interfèrent sur les définitions de certains caractères). Rappelons que 30 est le code du curseur, 31 le code de la bordure et 32 le code de l'espace.

Nouvelles fonctions

CALL LINK ("DISPLH", ligne, colonne, chaîne)

Affiche horizontalement la chaîne à partir de la ligne et de la colonne données.

1 <= ligne <= 24
1 <= colonne <= 32
Chaîne; exemple :

- "BONJOUR"
- A$
- B$(3) ou tout élément d'un tableau est interdit. Dans ce cas faire successivement A$=B$(3) suivit de CALL LINK ("DISPLH", 3, 23, A$).

La chaîne ne doit pas dépasser le bas de l'écran, c'est-à-dire 32L + C − 33 + LEN (A$) <= 768 et LEN (A$) <= 255.

CALL LINK ("DISPLV", ligne, colonne, chaîne)

Affiche verticalement la chaîne à partir de la ligne et de la colonne données.

1< = ligne <= 24
1< = colonne <= 32

Chaîne; exemple :

- "BONJOUR"
- A$
- B$(3) est interdit.

La chaîne ne doit pas dépasser le bas de l'écran, c'est-à-dire L + LEN(A$) <= 25 et LEN (A$) <= 255.

CALL LINK ("CHAR", numéro du premier caractère à redéfinir, chaîne)

Les 16 premiers éléments de la chaîne déterminent le patron du caractère N, les 16 suivants déterminent le patron du caractère N + 1, etc...

Le numéro du premier caractère à redéfinir peut aller de 1 à 159 (avec les réserves du préambule).
La chaîne ne doit contenir que des éléments de /0,1,...,9,A,B,C,D,E,F/ exemple : "1AB3C", A$ mais pas B$(2).
La longueur de la chaîne doit être un nombre pair.
On ne peut définir au delà du caractère numéro 159, c'est-à-dire N + INT (LEN (chaîne − 1 / 16) <= 159.

CALL LINK ("REP", nombre de répétitions, chaîne à répéter, chaîne de retour)

Renvoie dans la chaîne de retour la chaîne à répéter juxtaposée à elle-même, le nombre de fois indiqué.

Nombre de répétitions >= 2.
Chaîne à répéter sous forme "BONJOUR" ou A$, mais pas élément d'un tableau B$(1).
Chaîne de retour (C$, par exemple, reçoit le résultat).

La chaîne de retour doit avoir une longueur inférieure ou égale à 255.

CALL LINK ("ACCEPT", ligne, colonne, taille obligatoire, chaîne de retour)

Saisit une chaîne de longueur donnée par la taille à partir de la ligne et de la colonne données et met cette chaîne dans la chaîne de retour.

1 <= ligne <= 24
1 <= colonne <=32

Taille :

- si c'est un nombre positif, 4 par exemple, dispose 4 blancs sur l'écran et saisit 4 caractères;
- si c'est un nombre négatif, −7 par exemple, il n'y a pas effacement de caractères et on saisit 7 caractères.

Si on veut saisir un nombre, mettre à la ligne suivante du programme A = VAL (A$)

CALL LINK ("COLOR", 1er groupe de caractères, nombre de groupes consécutifs, couleur des caractères, couleur du fond)

Met aux couleurs indiquées un certain nombre de groupes de caractères à partir du premier groupe indiqué.

Le 1er groupe de caractères peut aller de −3 à 16 :

- groupe −3; caractères 0 à 7
- groupe −2; caractères 8 à 15
- groupe −1; caractères 16 à 23
- groupe 0; caractères 24 à 31

Nombre de groupes : à choisir pour ne pas dépasser le groupe 16. 1er groupe + Nombre de groupes <= 17.

Pour les couleurs du caractères et du fond, employer le code de couleur habituel, par exemple 7 pour rouge foncé.

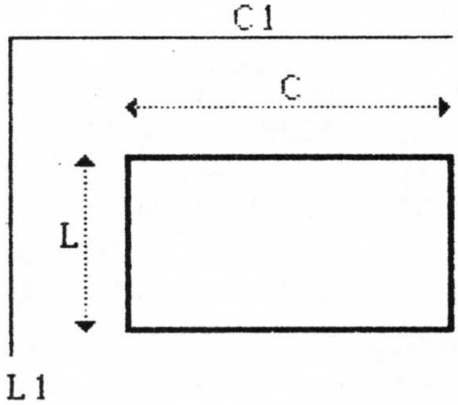
CALL LINK("SCROLL", ligne L1 du coin supérieur

gauche, colonne C1 du coin supérieur gauche, nombre de lignes L ,nombres de colonne C ,chaîne)

La chaîne s'inscrit à la ligne L1 + L − 1 à partir de la colonne C1, elle est complétée éventuellement par des blancs pour aller jusqu'à la colonne C1 + C − 1.

Ligne L1 du coin supérieur gauche de 1 à 24.
Colonne C1 du coin supérieur gauche de 1 à 32.
L entier tel que L1 + L <= 25.
C entier tel que C1 + C <= 33.
Chaîne telle que LEN (chaîne) <= C

La ligne L1, de la colonne C1 à la colonne C1 + C − 1, disparaît.
La ligne L1 + 1, de la colonne C1 + C − 1, passe à la ligne L1, mêmes colonnes.
La ligne L1 + L − 2, de la colonne C1 à la colonne C1 + C − 1, passe à la ligne du dessus.



Si vous désirez créer vous-même d'autres fonctions qui vous manquent en Basic, vous pouvez, par exemple, procéder de la façon suivante :

- charger le programme dans la Mini-mémoire, faire, avec *EASY BUG, S7118 TO 74EE;*
- charger votre assembleur ligne par ligne;
- créer vos fonctions à partir de l'adresse >7D00, par exemple faire *S 7D00 TO 8000;*
- charger votre premier enregistrement;
- créer la table REF/DEF;
- ajuster le contenu des adresses >701C, >701D, >701E et >701F;
- sauvegarder le tout avec *S7000 TO 8000.* ■

```
100 REM  *******************
110 REM  *     Exemples    *
120 REM  *  d'utilisation  *
130 REM  *  des fonctions  *
140 REM  *******************
150 REM  *  Basic TI plus  *
160 REM  *  Mini-memoire   *
170 REM  *******************
180 REM
190 CALL CLEAR
200 REM  ----CHOIX DE LA COULEUR DE L'ECR
AN -----
210 CALL LINK("DISPLH",2,3,"TAPEZ LE CODE
 DE LA COULEUR")
220 CALL LINK("DISPLH",3,3,"DE L'ECRAN PU
IS FAITES ENTER")
230 CALL LINK("ACCEPT",5,15,2,A$)
240 A=VAL(A$)
250 IF (A<2)+(A>16)THEN 230
260 CALL SCREEN(A)
270 REM   ------ CHOIX DES COULEURS -----
--
280 IF A=5 THEN 310
290 CALL LINK("COLOR",1,8,5,1)
300 GOTO 320
310 CALL LINK("COLOR",1,8,2,1)
320 IF A=7 THEN 350
330 CALL LINK("COLOR",9,2,7,15)
340 GOTO 370
350 CALL LINK("COLOR",9,2,13,10)
360 REM  --- MOULINS ------
370 D$="01010101010101FFFF010101010101018
0808080808080FFFF80808080808080"
380 CALL LINK("CHAR",97,D$)
390 CALL CHAR(104,"0")
400 CALL LINK("REP",25,"h",BORD$)
410 CALL LINK("REP",8,"hac",A$)
420 CALL LINK("REP",8,"hbd",B$)
430 CALL LINK("DISPLH",18,4,BORD$)
440 CALL LINK("DISPLH",21,4,BORD$)
450 CALL LINK("DISPLH",19,4,A$&"h")
460 CALL LINK("DISPLH",20,4,B$&"h")
470 E$="080C040602030101030E1C70C00000000
0000000030E38E080C0406020301018"
480 F$="C0E070381C0E070303070E1C3870E0C00
3070E1C3870E0C0C0E070381C0E0703"
490 G$="00000000C0701C0701010101030206040C0
8181030206040C0C0F01C0701000000"
500 CALL LINK("CHAR",97,D$)
510 CALL LINK("CHAR",97,E$)
520 CALL LINK("CHAR",97,F$)
530 CALL LINK("CHAR",97,G$)
540 CALL KEY(3,K,S)
550 IF S<>1 THEN 500
560 REM  ------- SCROLL -----
570 CALL CLEAR
580 CALL COLOR(10,A,A)
590 CALL HCHAR(1,1,104,768)
600 CALL LINK("COLOR",1,8,5,16)
610 CALL COLOR(2,2,2)
620 CALL HCHAR(4,4,40,26)
630 CALL LINK("REP",26,"(",A$)
640 CALL LINK("DISPLH",20,4,A$)
650 J=2
660 PAS=1
670 CALL LINK("REP",J," ",DEB$)
680 CALL LINK("SCROLL",5,5,15,24,DEB$&"BO
NJOUR")
```

```
690 CALL KEY(3,K,S)
700 IF S=1 THEN 770
710 J=J+PAS
720 IF (J<>16)*(J<>1)THEN 670
730 PAS=PAS*(-1)
740 J=J+2*PAS
750 GOTO 670
760 REM  DEFINITION DE CARACTERES DE CODE
 ASCII EVENTUELLEMENT INFERIEURS A 32
770 CALL CLEAR
780 CALL SCREEN(8)
790 CALL COLOR(2,2,16)
800 CALL LINK("DISPLH",10,3,"QUEL CARACTE
RE VOULEZ-VOUS    REDEFINIR ? ")
810 CALL LINK("ACCEPT",11,15,3,A$)
820 A=VAL(A$)
830 CALL LINK("DISPLH",14,3,"CODE DU PATR
ON ?                      FFFFFFFFF0000000
")
840 CALL LINK("ACCEPT",15,6,-16,B$)
850 CALL LINK("CHAR",A,B$)
860 C0=INT(A/8)-3
870 CALL LINK("COLOR",C0,1,5,12)
880 CALL LINK("REP",20,CHR$(A),C$)
890 CALL LINK("DISPLH",22,7,C$)
900 CALL KEY(3,K,S)
910 IF S=1 THEN 800
920 GOTO 900
```

```
        DEF   DISPLH,CHAR
        DEF   SCROLL,COLOR
        DEF   REP,ACCEPT
        DEF   DISPLV

        AORG  >7118

FI      CLR   @>837C
        LWPI  >83E0
        B     @>0070

RE      BLWP  @>6024
        JMP   L1
L2      MOVB  R1,@>8C00
L1      DEC   R2
        JNE   L2
        B     *R11

LN      CLR   R0
        BLWP  @>6044
        BLWP  @>601C
        DATA  >1200
        B     *R11

LC      CLR   R0
        LI    R2,TA
        SETO  *R2
        BLWP  @>604C
        CLR   R5
        MOVB  *R2+,R5
        SWPB  R5
        B     *R11

RC      CLR   R0
        LI    R2,TA
        BLWP  @>6048
        B     *R11

*       DISPLH
*       DISPLV

DI      LI    R1,1
        BL    @LN
        MOV   @>834A,R4
        SLA   R4,5
        INC   R1
        BL    @LN
        MOV   @>834A,R3
        A     R3,R4
        AI    R4,-33

        INC   R1
QQ      BL    @LC
        LI    R6,>6000
        MOV   R4,R0

BO      MOVB  *R2+,R1
        AB    R6,R1
        BLWP  @>6024
        A     R7,R0
        DEC   R5
        JNE   BO
        B     @FI

DISPLH  LI    R7,1
        B     @DI

DISPLV  LI    R7,32
        B     @DI

*       CHAR
*
CHAR    LI    R1,1
        BL    @LN
        MOV   @>834A,R4
        AI    R4,96
        SLA   R4,3
        INC   R1
        BL    @LC

        MOV   R4,R0
SU      MOVB  *R2+,R1
        BL    @TR
        SLA   R1,4
        MOV   R1,R4
        MOVB  *R2+,R1
        BL    @TR
        A     R4,R1
        SWPB  R1
        BLWP  @>6024
        INC   R0
        DECT  R5
        JGT   SU
        B     @FI

TR      SWPB  R1
        ANDI  R1,>00FF
        CI    R1,64
        JGT   L
        AI    R1,-48
        JMP   R
L       AI    R1,-55
R       B     *R11

*       REP

REP     LI    R1,1
        BL    @LN
```

```
        MOV   @>834A,R4
        DEC   R4
        INC   R1
        BL    @LC
        MOV   R5,R6
        LI    R0,TA+1
        MOV   R0,R1
B2      MOV   R1,R2
        A     R5,R2
        MOVB  *R1,*R2
        INC   R1
        DEC   R6
        JNE   B2

        SWPB  R5
        AB    R5,@TA
        SWPB  R5
        MOV   R5,R6
        DEC   R4
        JNE   B2
        LI    R1,3
        BL    @RC
        BL    @FI

*       ACCEPT

ACCEPT  LI    R1,1
        BL    @LN
        MOV   @>834A,R4
        SLA   R4,5
        INC   R1
        BL    @LN
        MOV   @>834A,R3
        A     R3,R4
        AI    R4,-33
        INC   R1
        BL    @LN
        MOV   @>834A,R5
        CI    R5,0
        JLT   S1
        BL    @N

S1      ABS   R5
        MOV   R4,R0
        MOV   R4,R8
        A     R5,R8

A       CLR   R1
        BLWP  @>602C
        MOV   R1,R6
SC      LI    R1,>7E00
        BLWP  @>6024
        CLR   @>8374
        BLWP  @>6020
        MOVB  @>837C,R10
        MOV   R6,R1
        BLWP  @>6024
        MOVB  R10,R10
        JEQ   SC
        MOVB  @>8375,R1
        ANDI  R1,>FF00

        CI    R1,>FF00
        JEQ   SC
        CI    R1,>0D00
        JEQ   T
        CI    R1,>0800
        JEQ   S
        SWPB  R1
        CI    R1,32
        JLT   SC
        CI    R1,126
        JGT   SC
        SWPB  R1
        AI    R1,>6000
        BLWP  @>6024
        INC   R0
        C     R0,R8
        JLT   A
        DEC   R0
        JMP   A

T       MOV   R5,R2
        MOV   R4,R0
        LI    R1,TA+1
        BLWP  @>6030

        MOV   R1,R5
        A     R2,R5
B3      DEC   R5
        CLR   R7
        MOVB  *R5,R7
        CI    R7,>8000
        JLE   BL
B4      AI    R7,>A000
        MOVB  R7,*R5
        DEC   R5
        MOVB  *R5,R7
        CI    R5,TA
        JNE   B4
        JMP   RR
BL      DEC   R2
        JNE   B3
RR      SWPB  R2
        MOVB  R2,@TA
        LI    R1,4
        CLR   @>837C
        BL    @RC
        B     @FI

N       MOV   R11,R10
        MOV   R4,R0
        LI    R1,>8000
        MOV   R5,R2
        BL    @RE
        B     *R10
S       C     R4,R0
        JEQ   SC
        DEC   R0
        JMP   A

*       COLOR

COLOR   LI    R1,1
        BL    @LN
        MOV   @>834A,R4
        AI    R4,783

        INC   R1
        BL    @LN
        MOV   @>834A,R2
        INC   R1
        BL    @LN
        MOV   @>834A,R3
        DEC   R3
        SLA   R3,4
        INC   R1
        BL    @LN
        MOV   @>834A,R1
        DEC   R1
        A     R3,R1
        SWPB  R1
        MOV   R4,R0
        BL    @RE
        B     @FI

*       SCROL

SCROLL  LI    R1,1
        BL    @LN
        MOV   @>834A,R4
        SLA   R4,5
        INC   R1
        BL    @LN
        MOV   @>834A,R3
        A     R3,R4
```

```
       AI    R4,-33
       INC   R1
       BL    @LN
       MOV   @>834A,R5
       DEC   R5
       INC   R1
       BL    @LN
       MOV   @>834A,R2
       MOV   R4,R0
       AI    R0,-32
```

```
       LI    R1,TA
       CLR   R6

Q      AI    R0,64
       BLWP  @>6030
       AI    R0,-32
       BLWP  @>6028
       INC   R6
       C     R6,R5
       JL    Q
```

```
       AI    R0,32
       MOV   R0,R4
       LI    R1,>8000
       BL    @RE

       LI    R1,5
       B     @QQ

TA     BSS   256
       END
```

Le Basic à la loupe

Marianne Sutz

Cet article vous propose un petit voyage dans la mémoire du TI-99/4A, afin de découvrir comment sont stockées les lignes constituant un programme écrit en Basic TI ou en Basic Etendu. Nous verrons qu'un programme peut très bien se modifier lui-même ou, pourquoi pas, générer d'autres programmes.

Les "tokens"

Le langage Basic est composé de mots clés qui sont relativement longs, du moins pour certains (RANDOMIZE, RESTORE...). Si ces instructions étaient conservées sans modification, la mémoire de notre ordinateur serait très rapidement saturée. Pour pallier en partie cet inconvénient - le Basic reste un "dévoreur de mémoire" - les mots clés sont codés et n'occupent qu'un seul octet appelé "token". Afin de faire la différence avec les codes ASCII standards, les tokens utilisent les octets hauts, c'est-à-dire ceux dont le bit de poids fort est mis à 1 (voir l'article sur les systèmes de numérations, publié dans le précédent numéro de 99 Magazine).

Vous trouverez, à la fin de cet article, une liste complète des tokens employés par Texas Instruments. Les instructions spécifiques au Basic Etendu sont soulignées et ne sont bien sûr pas reconnues par le Basic TI, ce qui explique les effets bizarres obtenus lorsque l'on tente de lister un programme écrit en Basic Etendu, sans le module du même nom.

La suite des explications ne comprendra que des exemples utilisant le système décimal; cependant, les tokens sont donnés en décimal et en hexadécimal afin de vous éviter la conversion lorsque vous utiliserez la liste pour vos propres applications

Les indicateurs

Indicateur de chaînes fermées par des guillemets

Lorsque l'interpréteur rencontre cet indicateur, qui est un octet contenant la valeur 199, il sait que ce qui suit est une chaîne de caractères représentant une constante alphanumérique. L'indicateur est suivi par un octet donnant la longueur de la chaîne, ainsi, "ABCDEFG" devient :

199 *(indicateur)* 7 *(longueur)* 65 66 67 68 69 70 71 *(chaîne)*.

Indicateur de chaînes non fermées

Cette fois, l'indicateur est un octet à 200 qui fait savoir à l'interpréteur qu'il va rencontrer une constante numérique ou un mot clé non tokenisé. Comme pour l'indicateur de constantes alphanumériques, la valeur 200 est suivi par un octet pour la longueur.

Puisque nous avons la liste des tokens, nous pouvons déjà "traduire" certaines choses; par exemple, CALL CLEAR devient, dans une ligne de Basic :

157 *(CALL)* 200 *(indicateur)* 5 *(longueur)* 67 *(C)* 76 *(L)* 69 *(E)* 65 *(A)* 82 *(R)*.

De même, A = 123.32 devient :

65 *(les noms de variables sont représentés par des codes ASCII standards)* 190 *(=)* 200 *(indicateur)* 6 *(longueur)* 49 50 51 46 *(point)* 51 50.

Indicateur de numéro de ligne

Il est égal à 201 et précède toujours un numéro de ligne. On le rencontre donc toujours après les instructions GOTO, THEN, etc... et RESTORE lorsqu'un numéro de ligne est employé.

Les numéros de lignes sont codés sur seize bits et en complément à deux (32767 maximum), ce qui fait que deux octets sont nécessaires à la représentation des numéros. GOTO 500 devient :

134 *(goto)* 201 *(indicateur)* 1 *(partie entière de 500/256)* 244 *(500 moins 256)*.

Constitution d'une ligne

Chaque ligne est précédée par un octet contenant la longueur totale de la ligne, et se termine par un octet nul (tous les bits sont à zéro). De cette façon, l'interpréteur sait qu'il est en fin de ligne dès qu'il rencontre un zéro, sauf lorsque cette valeur est précédée par un octet contenant 201 car, dans ce cas, il s'agit de l'octet de poids fort d'un numéro de ligne inférieur à 256.

De même, si l'octet à zéro est situé deux octets plus loin que l'indicateur "201", nous avons à faire à l'octet de poids faible d'un numéro de ligne multiple de 256. Par exemple, une hypothétique ligne 2560 serait représentée par 10 (poids fort) et 0 (poids faible) puisque 2560 est égal à 256 * 10 + 0.

Signalons que l'octet terminal est pris en compte par l'octet indiquant la longueur de la ligne, et que cette même ligne peut contenir jusqu'à 255 octets.

Nous sommes maintenant en possession de tous les éléments requis pour l'écriture d'une ligne de Basic à la manière de l'interpréteur. Par exemple, *FOR B=0 TO 10 STEP 2* devient, quand vous enfoncez la touche *ENTER* pour la valider :

16 *(longueur de la ligne)* 140 *(FOR)* 66 *(A)* 190 *(=)* 200 *(indicateur)* 1 *(longueur de chaîne)* 48 *(0)* 177 *(TO)* 200 *(indicateur)* 2 *(longueur)* 49 48 *(10)* 178 *(step)* 200 *(indicateur)* 1 *(longueur)* 50 *(2)* 0 *(octet terminal)*.

Position des lignes en mémoire

En Basic TI, les lignes sont toujours stockées dans la mémoire vive du processeur vidéo, même si une ex-

tension 32 Ko est raccordée au système. En Basic Etendu, les lignes sont en mémoire vidéo si l'extension n'est pas présente, alors qu'elles sont situées dans la mémoire vive directement adressable par le microprocesseur dans le cas contraire, et utilisent les adresses 40960 à 65535 (>A000 à >FFFF).

Dans tous les cas, les lignes sont stockées dans l'ordre dans lequel elles sont entrées (les numéros ne sont pas pris en compte), et ce depuis les adresses fortes vers les adresses faibles. Le dernier octet (octet terminal) de la première ligne entrée se trouve à l'adresse :

- 16383 (>3FFF) sans contrôleur de disquette;
- 14295 (>37D7) avec un contrôleur de disquette;
- 14813 (>39DD) avec un contrôleur de disquette et CALL FILES (2);
- 15531 (>3CAB) avec un contrôleur de disquette et CALL FILES (1).

Table des numéros de ligne

Chaque ligne du programme occupe quatre octets dans cette table située juste avant la dernière ligne entrée. Les deux premiers octets représentent le numéro de ligne alors que les deux suivants indiquent l'adresse en mémoire du premier octet de la ligne (il s'agit bien du premier octet de la ligne et non pas de l'octet qui indique la longueur).

Si l'on consulte le contenu de cette table, nous nous apercevons que les numéros de lignes apparaissent dans l'ordre, mais à l'envers. Ainsi, pour un petit programme constitué des lignes 10, 20 et 30, nous trouvons dans la table :

- 00 30 - adresse de la ligne 30;
- 00 20 - adresse de la ligne 20;
- 00 10 - adresse de la ligne 10.

L'adresse du début de la table des lignes (octet de poids fort du numéro de ligne le plus élevé) est située à l'adresse −31952 (>8330) en Basic Etendu et −31970 (>831E) en Basic TI.

Un petit programme

Le programme proposé, pour concrétiser ce que nous venons de voir, réalise son propre "décodage". Vous pouvez l'incorporer à vos programmes pour en étudier le contenu.

Pour chaque ligne analysée, vous obtenez l'affichage du numéro de ligne, de l'adresse en mémoire du premier octet de la ligne, du nombre d'octets la constituant et de son contenu en décimal et en hexadécimal. La première ligne traitée est la dernière ligne du programme. Pour visualiser la ligne suivante (dans la table des numéros de ligne, ligne précédente dans le programme), il faut appuyer sur une touche quelconque.

Très important : le programme utilise l'adresse −31970, qui contient l'adresse où se trouve l'octet de poids fort du numéro de la dernière ligne. Cette adresse n'est à jour qu'après l'utilisation de la commande "LIST"; il faut donc utiliser cette commande avant de faire démarrer le programme. Il est inutile de lister la totalité du programme; "LIST 1", par exemple, met à jour l'adresse qui nous intéresse.

Fonctionnement du programme

Les lignes 12 à 16 initialisent un tableau qui sera utilisé pour les conversions des nombres décimaux en nombres hexadécimaux.

La ligne 17 place l'octet de poids fort de l'adresse du premier numéro de ligne dans la variable A, alors que l'octet de poids faible est placé dans la variable B. L'adresse −31970 est directement adressable par le microprocesseur, il faut donc utiliser l'instruction *CALL PEEK*.

La petite opération de la ligne 18 nous permet d'obtenir une adresse réelle dans *ATL* (pour "Adresse de la Table des numéros de Lignes").

La variable *NLP* (pour "Numéro de la Ligne Précédente") sera utilisée pour savoir si l'on sort de la table de numéros de lignes.

Les lignes 20 et 21 vont chercher le numéro de la ligne et le placent dans la variable *NL*. Puisque la table des numéros de ligne est située dans la mémoire du processeur vidéo, nous utilisons l'instruction *CALL PEEKV*.

La ligne 22 nous permet de déterminer si la dernière ligne a été consultée. Voici pourquoi :

- juste après la table des numéros de ligne, nous trouvons le début de la dernière ligne entrée, soit l'octet qui indique la longueur de la ligne et le premier octet faisant réellement partie de la ligne; lorsque le programme "sortira" de la table des numéros, il considérera ces deux octets comme étant le numéro de la ligne suivante;
- une ligne est constituée d'au moins deux octets (une instruction sans argument, comme "END", et un octet terminal), nous sommes donc sûr que l'octet indiquant la longueur sera toujours supérieur ou égal à 2. Comme cette valeur est mutipliée par 256 (ligne 21), le "faux numéro de ligne" ne peut être inférieur à 512 + le token de l'instruction soit, dans le cas de l'instruction "PRINT", 512 + 156 = 668;
- le "faux numéro de ligne" survenant juste après le dernier numéro de ligne de la table, qui est le numéro de la première ligne du programme, si l'on admet que le numéro de la première ligne d'un programme est rarement inférieur à 668, le fait de trouver un numéro de ligne supérieur au numéro de la ligne précédente nous indique que toutes les lignes ont été traitées.

Pour le traitement de la première ligne, la variable *NLP* est initialisée à 65535 qui est une valeur impossible puisqu'un numéro de ligne ne peut être supérieur à 32767 (seize bits en complément à deux).Comme nous l'avons vu précédemment, les numéros de ligne sont suivis par deux octets constituant l'adresse du premier octet de la ligne; l'adresse du premier de ces deux octets est donc égale à ATL + 2.

Les lignes 25 et 26 agissent de la même façon que les lignes 20 et 21 mais retournent dans *APO* l'adresse du premier octet de la ligne en question.

L'octet précédent le premier octet de la ligne contenant le nombre d'octets dans la ligne, la ligne 28 retourne ce nombre dans *NO*.
APOH conserve le nombre d'octets pour l'affichage du contenu de la ligne en hexadécimal.

La ligne 31 retourne le contenu d'un des octet de la ligne. La variable *APO* sera ensuite incrémentée de un pour pointer l'octet suivant.
Si l'octet contient la valeur "zéro", il s'agit d'un octet terminal ou d'une partie d'une adresse précédée par l'indicateur "201". Dans ce cas, le programme est dirigé vers les lignes 36 et 37 où les deux octets placés avant l'octets concernée sont testés.
Si un de ces deux octets contient "201", le "décodage" de la ligne continue. Sinon, nous ajoutons 4 à la variable *ATL*, la variable *NLP* est mise à jour et nous passons à la ligne suivante.

En Basic Etendu

Pour écrire et utiliser ce type de programme avec le Basic Etendu, il faut faire quelques remarques :

- une extension de mémoire est indispensable puisque les instructions *CALL PEEK* et *CALL LOAD* ne sont pas utilisables sans cet accessoire;
- l'adresse du premier octet de la table des numéros de ligne ne se trouve plus à l'adresse −31970 mais à l'adresse −31952;

- lorsqu'une extension de mémoire est raccordée au TI, le Basic Etendu utilise la partie haute de l'extension pour le stockage de la table et des lignes du programme; il faut donc employer *CALL PEEK* au lieu de *CALL PEEKV* (de toutes façons, *CALL PEEKV* n'existe pas en Basic Etendu !);
- les adresses "peekées" dans −31952 et dans la table des numéros de ligne sont codées sur seize bits mais en valeurs non signées; elles peuvent donc être supérieures à 32767. Comme *CALL PEEK* n'admet pas les valeurs supérieures à 32767, il faut leur soustraire 65536 afin d'obtenir le complément à deux. Ainsi, 40000 devient 40000 − 65536 soit −25536.

Quelques constatations

Lorsque l'on regarde "à l'intérieur" d'un programme, on se rend compte que la mémoire de notre ordinateur est souvent encombrée inutilement : le programme accompagnant cet article à l'air tout à fait correct mais, si nous regardons d'un peu plus près...

Les REMarques

On est pour ou on est contre, là n'est pas la question. Par contre, si des espaces suivent la remarque ou l'instruction *REM*, ces espaces sont conservés. Cela est fort génant car on peut ainsi perdre beaucoup de précieux octets sans s'en apercevoir. Il faut aussi signaler que si l'instruction REM est employée seule (comme à la ligne 10 de notre programme), elle est toujours suivie d'un espace.

Les DATA

La ligne 13 occupe 65 octets ! C'est beaucoup pour seize caractères. Remplaçons les lignes 13 et 15 par :

13 D$ = "0123456789ABCDEF"
15 H$(B) = SEG$ (D$, B + 1, 1)

Regardez maintenant se qui ce trouve en mémoire, ou faîtes le calcul : nous avons gagné 30 octets sur deux lignes !

Les constantes

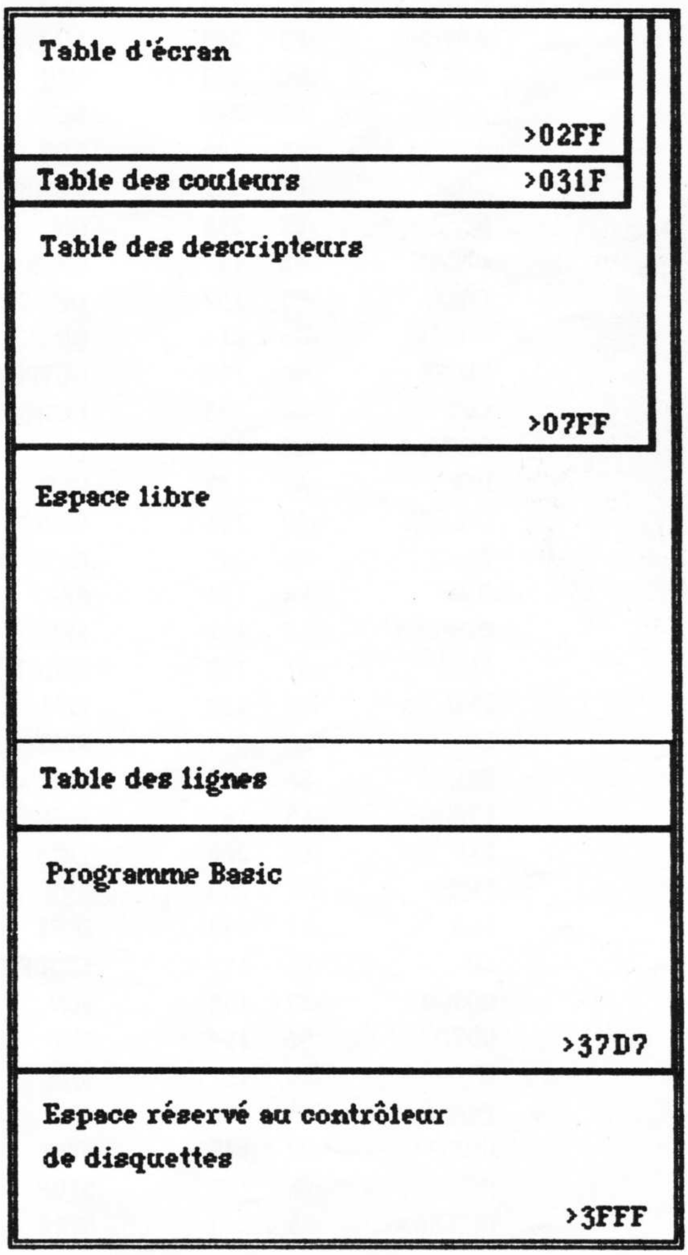
Dans tous les programmes, certaines valeurs sont utilisées fréquemment; c'est le cas de "1", utilisé six fois dans notre programme. Chaque constante est précédée par un indicateur (200) et un octet donnant la longueur de la chaîne; donc, notre "1" insignifiant occupe à chaque fois 3 octets alors qu'il aurait été préférable de donner la valeur "1" à une variable, étant donner qu'une variable d'un caractère occupe un octet. Ceci n'est bien sûr pas valable si l'on utilise des noms de variables "à rallonges".

Nous arrivons au terme de cet article mais le dossier est loin d'être clos. En effet, nous verrons une prochaine fois comment faire certaines opérations normalement impossibles (comme RESTORE A, RESTORE 10 ⋆ X, GOTO B + C...), comment sont stockées les variables, etc...

Si vous trouvez des astuces permettant de gagner de la place ou d'accélérer l'exécution des programmes, n'oubliez pas de nous le faire savoir. A la prochaine fois... ■

```
1 REM  **********************
2 REM  *    Ce programme    *
3 REM  *     fonctionne     *
4 REM  *  en Basic TI avec  *
5 REM  * la Mini-Memoire ou *
6 REM  *      le module     *
7 REM  * Editeur/Assembleur *
8 REM  * sans extension 32K *
9 REM  **********************
10 REM
11 CALL CLEAR
12 DIM H$(15)
13 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
14 FOR B=0 TO 15
15 READ H$(B)
16 NEXT B
17 CALL PEEK(-31970,A,B)
18 ATL=A*256+B
19 NLP=65535
20 CALL PEEKV(ATL,A,B)
21 NL=A*256+B
```

Utilisation de la mémoire vidéo avec le Basic et un contrôleur de disquettes



```
22 IF NL>NLP THEN 54
23 PRINT "LIGNE";NL
24 AL=ATL+2
25 CALL PEEKV(AL,A,B)
26 APO=A*256+B
27 PRINT "ADRESSE :";APO
28 CALL PEEKV(APO-1,NO)
29 PRINT STR$(NO);" OCTETS"::
30 APOH=APO
31 CALL PEEKV(APO,I)
32 PRINT I;
33 IF I=0 THEN 36
34 APO=APO+1
35 GOTO 31
36 CALL PEEKV(APO-2,X,Y)
37 IF (X=201)+(Y=201)THEN 34
38 PRINT ::
39 FOR B=1 TO NO
40 CALL PEEKV(APOH,I)
41 N1=INT(I/16)
42 N2=I-N1*16
43 PRINT " >"&H$(N1)&H$(N2);
44 APOH=APOH+1
45 NEXT B
46 ATL=ATL+4
47 NLP=NL
48 PRINT ::"____________________________"
::
49 CALL COLOR(8,10,1)
50 CALL KEY(0,K,S)
51 IF S=0 THEN 50
52 CALL COLOR(8,2,1)
53 GOTO 20
54 END
```

ABS	>CB	203	LEN	>D5	213	SUB	>A1	161
ACCEPT	>A4	164	LET	>8D	141	SUBEND	>A8	168
ALL	>EC	236	LINPUT	>AA	170	SUBEXIT	>A7	167
AND	>BB	187	LOG	>D0	208	TAB	>FC	252
APPEND	>F9	249	MAX	>DF	223	TAN	>D4	212
ASC	>DC	220	MIN	>E0	224	THEN	>B0	176
AT	>F0	240	NEXT	>96	150	TO	>B1	177
ATN	>CC	204	NOT	>BD	189	TRACE	>90	144
BASE	>F1	241	NUMERIC	>E8	232	UALPHA	>EA	234
BEEP	>EE	238	ON	>9B	155	UNBREAK	>8F	143
BREAK	>8E	142	OPEN	>9F	159	UNTRACE	>91	145
CALL	>9D	157	OPTION	>9E	158	UPDATE	>F8	248
CHR$	>D6	214	OR	>BA	186	USING	>ED	237
CLOSE	>A0	160	OUTPUT	>F7	247	VAL	>DA	218
COS	>CD	205	PERMANENT	>FB	251	VALIDATE	>FE	254
DATA	>93	147	PI	>DD	221	VARIABLE	>F3	243
DEF	>89	137	POS	>D9	217	WARNING	>A6	166
DELETE	>99	153	PRINT	>9C	156	XOR	>BC	188
DIGIT	>E9	233	RANDOMIZE	>95	149	! (exclam.)	>83	131
DIM	>8A	138	READ	>97	151	# (dièse)	>FD	253
DISPLAY	>A2	162	REC	>DE	222	&	>B8	184
ELSE	>81	129	RELATIVE	>F4	244	(	>B7	183
END	>8B	139	REM	>9A	154	)	>B6	182
EOF	>CA	202	RESTORE	>94	148	* (astérisq.)	>C3	195
ERASE	>EF	203	RETURN	>88	136	+ (plus)	>C1	193
ERROR	>A5	165	RND	>D7	215	, (virgule)	>B3	179
EXP	>CE	206	RPT$	>E1	225	- (moins)	>C2	194
FIXED	>FA	250	RUN	>A9	169	/	>C4	196
FOR	>8C	140	SEG$	>D8	216	:	>B5	181
GO	>85	133	SEQUENTIAL	>F6	246	:: (séparat.)	>82	130
GOSUB	>87	135	SGN	>D1	209	;	>B4	180
GOTO	>86	134	SIN	>D2	210	<	>BF	191
IF	>84	132	SIZE	>EB	235	= (égal)	>BE	190
IMAGE	>A3	163	SQR	>D3	211	>	>C0	192
INPUT	>92	146	STEP	>B2	178	^ (expon.)	>C5	197
INT	>CF	207	STOP	>98	152			
INTERNAL	>F5	245	STR$	>DB	219			

Star Trek

Jean-Philippe Guillemant

NDLR : ce programme écrit en **Basic Etendu** nécessite une extension 32 Ko et un lecteur de disquettes. Comme il n'est pas possible de stocker des programmes de plus de 8 Ko sur cassette, le programme a été enregistré, sur la cassette d'accompagnement de ce numéro, en trois parties. La première partie est autonome et doit donc être transférée sans modification.
La seconde et la troisième partie doivent être regroupées en utilisant la méthode suivante :

- charger la seconde partie en mémoire depuis la cassette ;
- sauvegarder ce segment sur disquette en le baptisant S1 ;
- charger la troisième partie en mémoire depuis la cassette ;
- sauvegarder ce segment sur disquette avec l'option MERGE (SAVE DSK 1.S2,MERGE) ;
- charger en mémoire le programme S1 depuis la disquette avec OLD DSK 1.S1 ;
- charger en mémoire le programme S2 depuis la disquette avec MERGE DSK1.S2 ;
- sauvegarder sur disquette le programme ainsi reconstitué avec SAVE DSK.1.STAR2. Le nom "STAR2" est impératif, car il y a un RUN "DSK 1.STAR2" dans la première partie ;
- détruire les programmes S1 et S2.

Les règles du jeu et les quelques commentaires insérés dans les programmes (le minimum compte tenu de la place en mémoire) sont suffisants pour comprendre le jeu. Seule peut-être une explication historique complémentaire est nécessaire en ce qui concerne la nomenclature galactique.

Comme le sait chaque écolier, la plupart des civilisations de la voie lactée ont donné des noms différents aux quadrants galactiques. Au cours de la troisième conférence mégallannique, toutes les cultures participantes se sont mises d'accord, en signant le document appelé depuis "Convention 2̂6", pour spécifier 64 subdivisions principales, selon une matrice 8∗8, dans toutes les représentations bi-dimensionnelles de la galaxie.
Ceci diffère partiellement de la culture Terrienne, qui a participé à l'organisation initiale de la Fédération Galactique, dont les cartes, vieilles de plus d'un siècle, ont toujours montré 16 régions principales nommées depuis le ciel de la terre. Ces régions étaient divisées en quatre "Quadrants" désignés par les anciens chiffres romains.

La relation entre la nomenclature galactique historique terrienne et la nomenclature galactique standard est montrée d'une part à la fin de la règle du jeu, d'autre part en librairie-ordinateur (option 6).

```
100 ! ********************
110 ! *      STARTREK      *
120 ! ********************
130 ! *  Premiere  partie  *
140 ! ********************
150 ! *    Basic Etendu    *
160 ! ********************
170 !
180 DATA 14
190 DATA 17
200 DATA 1,"INSTRUCTIONS POUR"
210 DATA 2,"""SUPER STAR TREK"""
220 DATA 3,"_________________"
230 DATA 5,"1. A la question `COMMANDE´,"
240 DATA 6,"frappez l'une des commandes"
250 DATA 7,"suivantes:"
260 DATA 9,"NAV - SRS - LRS - PHA - TOR"
270 DATA 10,"- SHE - DAM - COM - XXX -"
280 DATA 12,"2. Si vous tapez une comman-
"
290 DATA 13,"de invalide, une liste abr{-
"
300 DATA 14,"g{e des commandes autoris{es
"
310 DATA 15,"sera affich{e."
320 DATA 17,"3. Certaines commandes exi-"
330 DATA 18,"gent l'entr{e de param}tres.
"
340 DATA 19,"Si vous tapez un param}tre"
350 DATA 20,"invalide, la commande sera"
360 DATA 21,"annul{e."
370 DATA 17
380 DATA 1,"INSTRUCTIONS POUR"
390 DATA 2,"""SUPER STAR TREK"""
400 DATA 3,"_________________"
410 DATA 5,"4. Chaque message vous sera"
420 DATA 6,"transmis en r{p{titif au bas"
430 DATA 7,"de votre {cran r{cepteur."
440 DATA 8,"Quand vous l'aurez enregis-"
450 DATA 9,"tr{, vous pourrez taper sur"
460 DATA 10,"une touche quelconque pour"
470 DATA 11,"continuer votre mission."
480 DATA 13,"5. La galaxie est divis{e en
"
490 DATA 14,"un damier de 8*8 quadrants,"
500 DATA 15,"et chaque quadrant divis{"
510 DATA 16,"ensuite en une grille de"
520 DATA 17,"8*8 secteurs (conform{ment"
530 DATA 18,"@ la Convention Galactique"
540 DATA 19,"appel{e ""Convention 2^6"").
"
550 DATA 18
560 DATA 1,"INSTRUCTIONS POUR"
570 DATA 2,"""SUPER STAR TREK"""
580 DATA 3,"_________________"
590 DATA 5,"6. Il vous est fix{ un point"
```

```
600 DATA 6,"de d{part quelque part dans"
610 DATA 7,"la galaxie pour commencer"
620 DATA 8,"votre mission en tant que"
630 DATA 9,"Commandant du vaisseau stel-"
640 DATA 10,"laire:"
650 DATA 11,"U.S.S. ENTERPRISE."
660 DATA 13,"Votre mission est de rep{rer
"
670 DATA 14,"et de d{truire la flotte des
"
680 DATA 15,"vaisseaux de guerre KLINGONS
"
690 DATA 16,"qui menace la F{d{ration des
"
700 DATA 17,"Plan}tes Unies."
710 DATA 19,"7. Vous disposerez, en tant"
720 DATA 20,"que Commandant de bord, des"
730 DATA 21,"commandes suivantes:"
740 DATA 20
750 DATA 1,"*** NAV ***"
760 DATA 2,"Controle des moteurs WARP"
770 DATA 3,"____________________________"
780 DATA 5,"La route num{rique"
790 DATA 6,"correspond au        090"
800 DATA 7,"diagramme ci-   135  |  045"
810 DATA 8,"contre.           \  |  /"
820 DATA 9,"                   \ | /"
830 DATA 10,"La puissance        \|/"
840 DATA 11,"des moteurs  180____+____000
"
850 DATA 12,"WARP d{pend         /|\"
860 DATA 13,"de la distance     / | \"
870 DATA 14,"@ parcourir;      /  |  \"
880 DATA 15,"ainsi, pour    225  |  310"
890 DATA 16,"aller du qua-       270"
900 DATA 17,"drant 6-5, sec-"
910 DATA 18,"teur 2-3, au quadrant 4-5,"
920 DATA 19,"secteur 4-3, il vous faudra"
930 DATA 20,"prendre la route 90 avec un"
940 DATA 21,"facteur WARP de 1.8 ..."
950 DATA 18
960 DATA 1,"*** SRS ***"
970 DATA 2,"Exploration @ faible port{e"
980 DATA 3,"_______________________________"
990 DATA 5,"Vous montre le balayage de"
1000 DATA 6,"votre quadrant actuel."
1010 DATA 7,"Les symboles sur l'{cran de"
1020 DATA 8,"votre d{tecteur sont:"
1030 DATA 10," - E  Position de votre"
1040 DATA 11,"       Vaisseau Stellaire,"
1050 DATA 12," - K  Croiseur de bataille"
1060 DATA 13,"       KLINGON,"
1070 DATA 14," - B  Starbase f{d{rale"
1080 DATA 15,"       (ravitaillement, r{ar
-"
1090 DATA 16,"        mement, r{parations),
"
1100 DATA 17," - *  Etoile."
1110 DATA 19,"Un rapport de situation"
1120 DATA 20,"abr{g{ vous sera {galement"
1130 DATA 21,"transmis."
1140 DATA 19
1150 DATA 1,"*** LRS ***"
1160 DATA 2,"Exploration @ longue port{e"
1170 DATA 3,"______________________________"
1180 DATA 5,"D{crit l'espace dans les"
1190 DATA 6,"quadrants entourant l'ENTER-
"
1200 DATA 7,"PRISE (qui est au centre de"
1210 DATA 8,"la zone balay{e)."
1220 DATA 9,"Cet {tat est cod{ sous la"
1230 DATA 10,"forme `###', o~ le chiffre"
1240 DATA 11,"des unit{s indique le nombr
e"
1250 DATA 12,"d'{toiles, celui des dizai-
"
1260 DATA 13,"nes le nombre de Starbases,
"
1270 DATA 14,"et le chiffre des centaines
"
1280 DATA 15,"repr{sente le nombre de"
1290 DATA 16,"vaisseaux KLINGONS."
1300 DATA 18,"Par exemple, `207' indique
2"
1310 DATA 19,"KLINGONS, pas de Starbases"
1320 DATA 20,"et 7 {toiles dans le qua-"
1330 DATA 21,"drant concern{."
1340 DATA 12
1350 DATA 1,"*** PHA ***"
1360 DATA 2,"Controle des phasers"
1370 DATA 3,"_________________________"
1380 DATA 6,"Vous permet de d{truire les"
1390 DATA 7,"croiseurs de bataille des"
1400 DATA 8,"KLINGONS en les cinglant"
1410 DATA 9,"d'une quantit{ d'{nergie"
1420 DATA 10,"suffisante pour faire s'ef-
"
1430 DATA 11,"fondrer leur {cran de force
."
1440 DATA 14,"N'oubliez pas: les KLINGONS
"
1450 DATA 15,"poss}dent, eux aussi, des"
```

```
1460 DATA 16,"phasers !!!..."
1470 DATA 20
1480 DATA 1,"*** TOR ***"
1490 DATA 2,"Controle des torpilles"
1500 DATA 3,"photoniques"
1510 DATA 4,"____________"
1520 DATA 6,"La route torpille est celle"
1530 DATA 7,"d{finie par le controle des"
1540 DATA 8,"WARP's. Si le vaisseau KLIN-
"
1550 DATA 9,"GON est touch{, il est d{-"
1560 DATA 10,"truit et ne peut riposter."
1570 DATA 11,"Si vous le manquez, vous"
1580 DATA 12,"pouvez etre soumis au tir d
e"
1590 DATA 13,"ses phasers. De toute fa[on
,"
1600 DATA 14,"vous pouvez recevoir le tir
"
1610 DATA 15,"phaser de tous les autres"
1620 DATA 16,"KLINGONS pr{sents dans le"
1630 DATA 17,"quadrant."
1640 DATA 19,"La librairie-ordinateur pos
-"
1650 DATA 20,"s}de une option qui calcule
"
1660 DATA 21,"pour vous la trajectoire de
s"
1670 DATA 22,"torpilles."
1680 DATA 11
1690 DATA 1,"*** SHE ***"
1700 DATA 2,"Controle de l'{cran"
1710 DATA 3,"_______________"
1720 DATA 6,"D{finit le nombre d'unit{s"
1730 DATA 7,"d'{nergie affect{es aux"
1740 DATA 8,"{crans. L'{nergie est prise"
1750 DATA 9,"sur l'{nergie totale du"
1760 DATA 10,"Vaisseau."
1770 DATA 13,"Notez: l'affichage de l'{ta
t"
1780 DATA 14,"{nerg{tique total englobe"
1790 DATA 15,"l'{nergie des {crans."
1800 DATA 10
1810 DATA 1,"*** DAM ***"
1820 DATA 2,"Compte-rendu d'avaries"
1830 DATA 3,"__________________"
1840 DATA 6,"Donne l'{tat de r{paration"
1850 DATA 7,"des diff{rents {quipements"
1860 DATA 8,"de l'ENTERPRISE."
1870 DATA 11,"Un {tat de r{paration n{ga-
"
1880 DATA 12,"tif indique que l'installa-
"
1890 DATA 13,"tion est provisoirement hor
s"
1900 DATA 14,"service."
1910 DATA 18
1920 DATA 1,"*** COM ***"
1930 DATA 2,"Librairie-ordinateur"
1940 DATA 3,"____________________"
1950 DATA 5,"La librairie-ordinateur con-
"
1960 DATA 6,"tient 7 options:"
1970 DATA 8,"Option 0: Rappel des options
"
1980 DATA 9,"   disponibles."
1990 DATA 11,"Option 1: Enregistrement"
2000 DATA 12,"    galactique cumul{. Montr
e"
2010 DATA 13,"    l'enregistrement des r{-
"
2020 DATA 14,"    sultats de toutes les ex
-"
2030 DATA 15,"    plorations pass{es par"
2040 DATA 16,"    senseurs @ courte et lon
-"
2050 DATA 17,"    gue port{e."
2060 DATA 19,"Option 2: Rapport de situa-
"
2070 DATA 20,"    tion. Donne la Stardate,
"
2080 DATA 21,"    les KLINGONS et les Star
-"
2090 DATA 22,"    bases restant en jeu."
2100 DATA 15
2110 DATA 1,"*** COM ***"
2120 DATA 2,"Librairie-ordinateur"
2130 DATA 3,"____________________"
2140 DATA 5,"Option 3: Donn{es pour les"
2150 DATA 6,"   torpilles photoniques."
2160 DATA 7,"   Calcule les directions et
"
2170 DATA 8,"   distances de l'ENTERPRISE
"
2180 DATA 9,"    vers tous les KLINGONS du
"
2190 DATA 10,"    quadrant."
2200 DATA 12,"Option 4: Donn{es de naviga
-"
2210 DATA 13,"    tion vers les Starbases.
"
2220 DATA 14,"    Calcule la direction et"
```

```
2230 DATA 15,"    la distance de l'ENTER-"
2240 DATA 16,"    PRISE vers la Starbase"
2250 DATA 17,"    du quadrant actuel."
2260 DATA 18
2270 DATA 1,"*** COM ***"
2280 DATA 2,"Librairie-ordinateur"
2290 DATA 3,"____________________"
2300 DATA 5,"Option 5: Calculs de direc-"
2310 DATA 6,"   tion-distance. Vous per-"
2320 DATA 7,"   met d'entrer les coordon-
"
2330 DATA 8,"    n{es n{cessaires aux cal-
"
2340 DATA 9,"   culs de direction et de"
2350 DATA 10,"   distance."
2360 DATA 12,"Option 6: Carte renseign{e"
2370 DATA 13,"   de la galaxie. Indique l
a"
2380 DATA 14,"   relation entre les coor-
"
2390 DATA 15,"   donn{es Galactiques Car-
"
2400 DATA 16,"   t{siennes et les princi-
"
2410 DATA 17,"   pales r{gions de la"
2420 DATA 18,"   Galaxie mentionn{es dans
"
2430 DATA 19,"   le jeu (ainsi que montr{
"
2440 DATA 20,"   ci-apr}s)."
2450 DATA 21
2460 DATA 1,"NOMENCLATURE GALACTIQUE"
2470 DATA 2,"_______________________"
2480 DATA 4,"   1  2  3  4    5  6  7  8"
2490 DATA 5,"   ____________ ____________"
2500 DATA 6,"1|  ANTARES   |   SIRIUS   |
"
2510 DATA 7," | I II III IV| I II III IV|
"
2520 DATA 8,"2|   RIGEL    |    DENEB   |
"
2530 DATA 9," | I II III IV| I II III IV|
"
2540 DATA 10,"3|  PROCYON   |   CAPELLA
|"
2550 DATA 11," | I II III IV| I II III IV
|"
2560 DATA 12,"4|    VEGA    | BETELGEUSE
|"
2570 DATA 13," | I II III IV| I II III IV
|"
2580 DATA 14,"5|  CANOPUS   | ALDEBARAN
|"
2590 DATA 15," | I II III IV| I II III IV
|"
2600 DATA 16,"6|   ALTAIR   |  REGULUS
|"
2610 DATA 17," | I II III IV| I II III IV
|"
2620 DATA 18,"7| SAGITTAIRE |  ARCTURUS
|"
2630 DATA 19," | I II III IV| I II III IV
|"
2640 DATA 20,"8|   POLLUX   |    L'EPI
|"
2650 DATA 21," | I II III IV| I II III IV
|"
2660 DATA 22,"   ____________ ____________
"
2670 CALL CLEAR :: CALL MINUS :: GOSUB 28
00
2680 DISPLAY AT(22,1):"Voulez-vous les r}
gles du":"jeu (O,N):"
2690 ACCEPT AT(23,12)VALIDATE("ONon")SIZE
(-1)BEEP:YN$
2700 IF YN$="o" OR YN$="n" THEN DISPLAY A
T(24,1)BEEP:"Passez en ALPHA LOCK s.v.p
." :: GOTO 2690
2710 IF YN$="N" THEN 2790 ELSE IF YN$<>"O
" THEN 2690
2720 RESTORE 180 :: READ NBR
2730 FOR J=1 TO NBR :: CALL CLEAR :: CALL
 SCREEN(13):: FOR ZC=0 TO 12 :: CALL CO
LOR(ZC,1,1):: NEXT ZC :: READ AFF
2740 FOR I=1 TO AFF :: READ ROW,A$ :: DIS
PLAY AT(ROW,1):A$ :: NEXT I
2750 DISPLAY AT(1,26):J :: DISPLAY AT(23,
1)BEEP:RPT$("_",28);"TAPEZ `ENTER' POUR
 CONTINUER"
2760 FOR ZC=0 TO 12 :: CALL COLOR(ZC,12,1
):: NEXT ZC
2770 CALL KEY(0,K,S):: IF S=0 THEN 2770
2780 NEXT J :: CALL CLEAR
2790 RUN "DSK1.STAR2"
2800 DISPLAY AT(6,9):RPT$("_",11):TAB(8);
"< S U P E R >":TAB(9);RPT$("_",11)
2810 DISPLAY AT(11,5):RPT$("_",19):TAB(4)
;"< S T A R   T R E K >":TAB(5);RPT$("_
",19):: RETURN
2820 SUB MINUS
2830 DATA 37,00444C18306444
2840 DATA 39,0008102
```

```
2850 DATA 43,10101010FF10101
2860 DATA 47,010204081020408
2870 DATA 64,20100038484834
2880 DATA 91,00000384040381
2890 DATA 92,804020100804020l
2900 DATA 95,00000000FF
2910 DATA 96,00100804
2920 DATA 97,00000038484834
2930 DATA 98,0040407048483
2940 DATA 99,00000038404038
2950 DATA 100,00080838484834
2960 DATA 101,000018243C2018
2970 DATA 102,0018282038202
2980 DATA 103,000038484838083
2990 DATA 104,00202038242424
3000 DATA 105,00100030101038
3010 DATA 106,001000301010502
3020 DATA 107,00404050605048
3030 DATA 108,00301010101038
3040 DATA 109,0000006C545454
3050 DATA 110,00000078484848
3060 DATA 111,0000003048483
3070 DATA 112,000000704848704
3080 DATA 113,00000038484C3808
3090 DATA 114,0000002830202
3100 DATA 115,0000182038083
3110 DATA 116,00101038101018
3120 DATA 117,00000048484834
3130 DATA 118,0000002424281
3140 DATA 119,00000044545428
3150 DATA 120,00000028102828
3160 DATA 121,000000484830102
3170 DATA 122,00000038102038
3180 DATA 123,1020304878403
3190 DATA 124,101010101010101
3200 DATA 125,2010304878403
3210 DATA 126,20100048484834
3220 DATA 0
3230 RESTORE 2830
3240 READ Z0 :: IF Z0<>0 THEN READ Z$ ::
CALL CHAR(Z0,Z$):: GOTO 3240
3250 SUBEND

100 ! **********************
110 ! *       STARTREK      *
120 ! **********************
130 ! *   Seconde Partie    *
140 ! **********************
150 ! *    Basic Etendu     *
160 ! **********************
170 STOP
180 OPTION BASE 1
190 RANDOMIZE
200 ON WARNING NEXT
210 !================
220 !=initialisations=
230 !================
240 !ecran
250 CALL CHAR(128,"000000FFFF"):: CALL CO
LOR(13,13,8):: CALL CHARPAT(95,A$):: IF
 A$="00000000FF000000" THEN 320
260 CALL CHAR(64,"00000038447C4444"):: CA
LL CHAR(91,"0000003C4040403C"):: CALL C
HAR(95,"00000000FF")
270 CALL CHAR(123,"0000007C4078407C101010
10101010100000007C4078407C")
280 DISPLAY AT(7,9)ERASE ALL:RPT$("_",11)
:TAB(8);"< S U P E R >":TAB(9);RPT$("_"
,11)
290 DISPLAY AT(11,5):RPT$("_",19):TAB(4);
"< S T A R   T R E K >":TAB(5);RPT$("_"
,19)
300 DISPLAY AT(16,5):"PATIENTEZ UN INSTAN
T"
310 !variables
320 DIM G(8,8),K(3,3),Z(8,8),D(8)
330 !G(8,8)=description des quadrants;Z(8
,8)=quadrants visualises
340 !K(3,3)=description des klingons du q
uadrant:I = 1,2,3 pour chaque klingon;
J=1:ordonnee; J=2:abscisse; J=3:energie
350 !D(8)=etat equipements
360 LIG=11 !ligne d'affichage
370 VIT=1 !vitesse des messages
380 A1$="NAVSRSLRSPHATORDAMSHECOMXXX"
390 T,T0=INT(RND*20+2100)!T0=date debut;
T=date actuelle
400 T9=25+INT(RND*10)!T9=duree de jeu
410 E,E0=3000 !E0=energie totale; E=energ
ie actuelle
420 D0=0 !D0=indicateur d'entree en radou
b
430 P,P0=10 !P0=torpilles totales; P=torp
illes actuelles
440 S=0 !S=energie ecran
450 FOR I=1 TO 8 :: D(I)=0 :: FOR J=1 TO
8 :: Z(I,J)=0 :: R1=RND :: K3=0
460 IF R1>.98 THEN K3=K3+1
470 IF R1>.95 THEN K3=K3+1
480 IF R1>.80 THEN K3=K3+1
490 K9=K9+K3 :: B3=0 :: IF RND>.96 THEN B
3=1 :: B9=B9+1
```

```
500 G(I,J)=K3*100+B3*10+INT(RND*8+1):: NE
XT J :: NEXT I :: IF K9)T9 THEN T9=K9+1
510 Q1=INT(RND*8+1):: Q2=INT(RND*8+1):: S
1=INT(RND*8+1):: S2=INT(RND*8+1)
520 !Q1,S1=ordonnee Enterprise
530 !Q2,S2=abscisse Enterprise
540 IF B9()0 THEN 600
550 IF G(Q1,Q2)(200 THEN G(Q1,Q2)=G(Q1,Q2
)+100 :: K9=K9+1
560 B9=1 :: G(Q1,Q2)=G(Q1,Q2)+10 :: Q1=IN
T(RND*8+1):: Q2=INT(RND*8+1)
570 !=============
580 !=fin d'init.=
590 !=============
600 K7=K9 :: FOR I=1 TO 12 :: CALL COLOR(
I,1,1):: NEXT I :: IF B9=1 THEN X$="" E
LSE X$="s"
610 DISPLAY AT(2,1)BEEP ERASE ALL:"VOICI
VOS ORDRES":RPT$("_",16)
620 DISPLAY AT(5,3):"D{truisez les";K9;"v
aisseaux":"  de guerre KLINGON qui ont"
:"  envahi la Galaxie avant"
630 DISPLAY AT(8,3):"qu'ils n'attaquent l
e Q.G.":"  de la F{d{ration @ la":"  St
ardate";T0+T9;"; ceci vous"
640 DISPLAY AT(11,3):"laisse";T9;"jours.
Il y a";B9:"  Starbase";X$;" dans la Ga
laxie":"  pour ravitailler le Vais-"
650 DISPLAY AT(14,3):"seau":"":TAB(7);"U.
S.S. ENTERPRISE" :: CALL HCHAR(18,3,128
,28)
660 DISPLAY AT(20,1):"Etes-vous pret @ pr
endre le":"Commandement? O" :: CALL HCH
AR(23,3,128,28)
670 CALL HCHAR(1,3,128,28):: FOR I=1 TO 1
2 :: CALL COLOR(I,2,1):: NEXT I :: CALL
 COLOR(3,9,1,4,9,1)
680 ACCEPT AT(21,15)VALIDATE("O")SIZE(-1)
:REP$
690 CALL CLEAR :: CALL COLOR(3,2,1,4,2,1)
:: CALL HCHAR(10,3,128,28)
700 D4=RND/2 :: Z(Q1,Q2)=G(Q1,Q2):: SRS=1
710 IF T0()T THEN CALL RAZ(1,9):: DISPLAY
 AT(2,1):"Vous entrez dans le quadrant"
 :: GOTO 730
720 DISPLAY AT(1,1):"Votre mission commen
ce avec":"le vaisseau dans le quadrant"
730 CALL SITU(Q1,Q2,G$):: DISPLAY AT(3,1)
BEEP:"galactique:":TAB(14-LEN(G$)/2);G$
740 K3=INT(G(Q1,Q2)/100):: B3=INT(G(Q1,Q2
)/10)-10*K3 :: S3=G(Q1,Q2)-100*K3-10*B3
 :: CALL COND(K3,S)
750 !mise en place Enterprise
760 Q$=RPT$(" ",64):: CALL SETSECT(S1,S2,
Q$,"E")
770 !mise en place klingons
780 FOR I=1 TO 3 :: FOR J=1 TO 3 :: K(I,J
)=0 :: NEXT J :: NEXT I :: IF K3=0 THEN
 820
790 FOR I=1 TO K3 :: CALL ALLOC(K(I,1),K(
I,2),Q$):: CALL SETSECT(K(I,1),K(I,2),Q
$,"K")
800 K(I,3)=INT(200*(.5+RND)):: NEXT I
810 !mise en place starbase
820 IF B3()0 THEN CALL ALLOC(B4,B5,Q$)::
CALL SETSECT(B4,B5,Q$,"B")
830 !mise en place etoiles
840 FOR I=1 TO S3 :: CALL ALLOC(R1,R2,Q$)
:: CALL SETSECT(R1,R2,Q$,"*"):: NEXT I
850 IF SRS=0 THEN 870
860 GOSUB 2980
870 IF S+E)10 THEN IF E)10 OR D(7)=0 THEN
 940
880 TXT$="*** ERREUR FATALE ***  Vous ven
ez d'echouer votre Vaisseau dans l'Espa
ce. L'{nergie est trop faible"
890 TXT$=TXT$&" pour maneuvrer et le cont
role d'{cran ne peut commuter vers la s
alle propulseurs ..."
900 CALL BANDE(TXT$,VIT):: GOTO 2860
910 !====================
920 !=boucle de commande=
930 !====================
940 CALL RAZ(24,24):: DISPLAY AT(23,1):"V
OTRE COMMANDE:"
950 ACCEPT AT(23,17)VALIDATE(UALPHA)SIZE(
-3)BEEP:REP$ :: IF LEN(REP$)()3 THEN 10
00
960 I=INT((POS(A1$,REP$,1)+2)/3):: ON I+1
 GOTO 1000,1070,860,1680,1800,2040,2460
,2350,3220,2860
970 !===================
980 !=commande invalide=
990 !===================
1000 TXT$="Entrez l'une des commandes sui
vantes: "
1010 TXT$=TXT$&"... NAV  ...  SRS  ...  L
RS  ...  PHA  ...  TOR  ...  DAM  ...
SHE  ...  COM  ...  XXX  ..."
1020 CALL BANDE(TXT$,VIT):: GOTO 870
1030 !===========
1040 !=calculs de=
1050 !=navigation=
1060 !===========
```

```
1070 DISPLAY AT(24,1):"Route (0-360):"
1080 ACCEPT AT(24,16)VALIDATE(DIGIT)SIZE(
3)BEEP:C1 :: IF C1<=360 THEN 1100
1090 TXT$="Erreur dans les donn{es de nav
igation, COMMANDANT ..." :: CALL BANDE(
TXT$,VIT):: GOTO 870
1100 IF D(1)<0 THEN X$="0.2" ELSE X$="8"
1110 DISPLAY AT(24,1):"Facteur WARP (0-";
X$;"):" :: ACCEPT AT(24,20+LEN(X$))VALI
DATE(NUMERIC)BEEP:W1
1120 ON SGN(W1)+2 GOTO 1090,870,1130
1130 IF D(1)>=0 THEN 1160 ELSE IF W1<=.2
THEN 1160
1140 TXT$="Avarie des moteurs WARP. La vi
tesse maximum est WARP 0.2 ..." :: CALL
 BANDE(TXT$,VIT):: GOTO 870
1150 !parametres de nav O.K.
1160 NN=INT(W1*8+.5):: IF E-NN>=0 THEN 12
20
1170 TXT$="`SALLE MACHINES': Energie disp
onible insuffisante pour maneuvrer @ WA
RP "&STR$(W1)&" ..."
1180 CALL BANDE(TXT$,VIT):: IF S<NN-E OR
D(7)<0 THEN 870
1190 TXT$="`CONTROLE DEFLECTEURS' rend co
mpte: "&STR$(S)&" unit{s {nerg{tiques a
ctuellement affect{es "
1200 TXT$=TXT$&"aux {crans ..." :: CALL B
ANDE(TXT$,VIT):: GOTO 870
1210 !mouvement des klingons
1220 FOR I=1 TO 3 :: IF K(I,3)=0 THEN 124
0
1230 CALL SETSECT(K(I,1),K(I,2),Q$," "):: 
 CALL ALLOC(K(I,1),K(I,2),Q$):: CALL SE
TSECT(K(I,1),K(I,2),Q$,"K")
1240 NEXT I :: GOSUB 2650
1250 !gestion des avaries existantes
1260 IF W1>=1 THEN D6=1 ELSE D6=W1
1270 D1=0 :: FOR I=1 TO 8 :: IF D(I)>=0 T
HEN 1320
1280 D(I)=D(I)+D6 :: IF D(I)>-.1 AND D(I)
<0 THEN D(I)=-.1
1290 IF D(I)<0 THEN 1320
1300 D1=D1+1 :: IF D1=1 THEN TXT$="`CONTR
OLE AVARIES' rend compte: R{parations "
 ELSE TXT$=TXT$&", "
1310 CALL MATER(I,G$):: TXT$=TXT$&G$
1320 NEXT I :: IF D1<>0 THEN TXT$=TXT$&"
termin{es ..." :: CALL BANDE(TXT$,VIT)
1330 !nouvelles avaries
1340 IF RND>.2 THEN 1400
1350 R1=INT(RND*8+1):: TXT$="`CONTROLE AV
ARIES' rend compte: " :: CALL MATER(R1,
G$):: IF RND>=.6 THEN 1370
1360 D(R1)=D(R1)-(RND*5+1):: TXT$=TXT$&G$
&" endommag{ ..." :: GOTO 1380
1370 D(R1)=D(R1)+RND*3+1 :: TXT$=TXT$&"V{
rifications "&G$&" effectu{es ..."
1380 CALL BANDE(TXT$,VIT)
1390 !navigation
1400 CALL SETSECT(S1,S2,Q$," ")
1410 X=-SIN(C1*PI/180):: Y=COS(C1*PI/180)
:: Q4=Q1 :: Q5=Q2 :: X5=0
1420 FOR I=1 TO NN :: S1=S1+X :: S2=S2+Y
1430 IF INT(S1+.5)<1 THEN IF Q1=1 THEN X5
=1 :: S1=S1-X ELSE X5=2 :: Q1=Q1-1 :: S
1=S1+8
1440 IF INT(S1+.5)>8 THEN IF Q1=8 THEN X5
=1 :: S1=S1-X ELSE X5=2 :: Q1=Q1+1 :: S
1=S1-8
1450 IF INT(S2+.5)<1 THEN IF Q2=1 THEN X5
=1 :: S2=S2-Y ELSE X5=2 :: Q2=Q2-1 :: S
2=S2+8
1460 IF INT(S2+.5)>8 THEN IF Q2=8 THEN X5
=1 :: S2=S2-Y ELSE X5=2 :: Q2=Q2+1 :: S
2=S2-8
1470 !DISPLAY AT(24,1)BEEP:"Quadrant";Q1;
Q2;"/ Secteur";INT(S1+.5);INT(S2+.5)::
CALL DELAI
1480 ON X5+1 GOTO 1530,1490,1570
1490 TXT$="`SALLE RADIO' rend compte mess
age COMMANDANT Flotte Stellaire: Autori
sation franchir limites "
1500 TXT$=TXT$&"Galaxie est express{ment
refus{e. Coupez vos moteurs ..." :: CAL
L BANDE(TXT$,VIT)
1510 TXT$="`SALLE MACHINES' rend compte:
Moteurs WARP coup{s dans le secteur "
1520 TXT$=TXT$&STR$(INT(S1+.5))&","&STR$(
INT(S2+.5))&" du quadrant "&STR$(Q1)&",
"&STR$(Q2)&" ..." :: GOTO 1560
1530 CALL TEST(INT(S1+.5),INT(S2+.5),Q$,"
 ",Z3):: IF Z3=1 THEN 1570
1540 S1=S1-X :: S2=S2-Y :: TXT$="Moteurs
WARP arr{t{s dans le secteur "&STR$(INT
(S1+.5))&" , "
1550 TXT$=TXT$&STR$(INT(S2+.5))&" cause m
auvaise navigation ..."
1560 CALL BANDE(TXT$,VIT):: I=NN
1570 NEXT I :: S1=INT(S1+.5):: S2=INT(S2+
.5)
1580 IF W1>=1 THEN T=T+1 ELSE T=T+INT(10*
W1)/10
1590 IF T>T0+T9 THEN 2860
```

```
1600 E=E-NN-10 :: IF E>=0 THEN 1630
1610 TXT$="`CONTROLE ECRANS' fournit {ner
gie n{cessaire pour achever la maneuvre
 ..." :: CALL BANDE(TXT$,VIT)
1620 S=S+E :: E=0 :: IF S<=0 THEN S=0
1630 IF 8*Q1+Q2<>8*Q4+Q5 THEN 700 ELSE CA
LL SETSECT(S1,S2,Q$,"E"):: GOTO 850
1640 !==================
1650 !=detection longue=
1660 !=    distance    =
1670 !==================
1680 IF D(3)<0 THEN TXT$="DETECTEURS LONG
UE PORTEE ***hors service***" :: CALL B
ANDE(TXT$,VIT):: GOTO 870
1690 SRS=0 :: CALL RAZ(LIG,LIG+11)
1700 DISPLAY AT(LIG,2):"EXPLORATION LONGU
E DISTANCE":" A PARTIR DU QUADRANT";Q1;
",";Q2:" ";RPT$("_",26)
1710 FOR J=Q2-1 TO Q2+1 :: IF J>0 AND J<9
 THEN DISPLAY AT(LIG+3,6*(J-Q2+2)+1):J
1720 NEXT J :: DISPLAY AT(LIG+4,5):RPT$("
_",19)
1730 FOR I=Q1-1 TO Q1+1 :: X=LIG+3+(I-Q1+
2)*2 :: IF I>0 AND I<9 THEN DISPLAY AT(
X,3):I
1740 FOR J=Q2-1 TO Q2+1 :: Y=6*(J-Q2+2)-1
1750 IF I<1 OR I>8 OR J<1 OR J>8 THEN A$=
"***" ELSE Z(I,J)=G(I,J):: A$=SEG$(STR$
(G(I,J)+1000),2,3)
1760 DISPLAY AT(X,Y):"| "&A$ :: NEXT J ::
 DISPLAY AT(X,23):"|":TAB(5);RPT$("_",1
9):: NEXT I :: GOTO 850
1770 !=================
1780 !=tir des phasers=
1790 !=================
1800 IF D(4)<0 THEN TXT$="Controle PHASER
S ***hors service***" :: GOTO 1820 ELSE
 IF K3>0 THEN 1830
1810 TXT$="`RESP. SCIENT.' rend compte: P
as de vaisseaux ennemis d{tect{s dans c
e quadrant ..."
1820 CALL BANDE(TXT$,VIT):: GOTO 870
1830 IF D(8)<0 THEN TXT$="Avarie ordinate
ur diminue pr{cision ..." :: CALL BANDE
(TXT$,VIT)
1840 TXT$="PHASERS verrouill{s sur l'obje
ctif ..." :: CALL BANDE(TXT$,VIT)
1850 TXT$="L'{nergie disponible est de "&
STR$(E)&" unit{s ..." :: CALL BANDE(TXT
$,VIT)
1860 DISPLAY AT(23,1):"Nombre d'unit{s:"
 :: ACCEPT AT(23,18)VALIDATE(DIGIT)BEEP:
X :: IF X=0 THEN 870
1870 IF E-X<0 THEN 1850
1880 E=E-X :: IF D(7)<0 THEN X=X*RND
1890 X=INT(X/K3):: FOR I=1 TO 3 :: IF K(I
,3)<=0 THEN 1990
1900 H=INT(X/SQR((K(I,1)-S1)^2+(K(I,2)-S2
)^2)*(RND+2)):: IF H>.15*K(I,3)THEN 193
0
1910 TXT$="SENSEURS: pas d'avaries sur l'
ennemi en "&STR$(K(I,1))&" , "&STR$(K(I
,2))&" ..."
1920 CALL BANDE(TXT$,VIT):: GOTO 1990
1930 K(I,3)=K(I,3)-H
1940 TXT$=STR$(H)&" unit{s frappent le KL
INGON du secteur "&STR$(K(I,1))&" , "&S
TR$(K(I,2))&" ..." :: CALL BANDE(TXT$,VIT
)
1950 IF K(I,3)<=0 THEN TXT$="*** KLINGON
d{truit ***" :: CALL BANDE(TXT$,VIT)::
GOTO 1970
1960 TXT$="Les senseurs montrent qu'il lu
i reste "&STR$(K(I,3))&" unit{s ..." ::
 CALL BANDE(TXT$,VIT):: GOTO 1990
1970 K3=K3-1 :: K9=K9-1 :: CALL SETSECT(K
(I,1),K(I,2),Q$," ")
1980 K(I,3)=0 :: G(Q1,Q2)=G(Q1,Q2)-100 ::
 Z(Q1,Q2)=G(Q1,Q2):: IF K9<=0 THEN 2800
1990 NEXT I :: GOSUB 2650 :: CALL COND(K3
,S):: GOTO 850
2000 !=============
2010 != torpilles =
2020 !=photoniques=
2030 !=============
2040 IF P<=0 THEN TXT$="TORPILLES PHOTONI
QUES toutes d{pens{es ..." :: CALL BAND
E(TXT$,VIT):: GOTO 870
2050 IF D(5)<0 THEN TXT$="Tubes photoniqu
es ***hors service***" :: CALL BANDE(TX
T$,VIT):: GOTO 870
2060 DISPLAY AT(24,1):"Route Torpille (0-
360):"
2070 ACCEPT AT(24,24)VALIDATE(DIGIT)SIZE(
3)BEEP:C1 :: IF C1<=360 THEN 2090
2080 TXT$="Param}tre non valable, Command
ant ! ..." :: CALL BANDE(TXT$,VIT):: GO
TO 870
2090 X1=-SIN(C1*PI/180):: X2=COS(C1*PI/18
0):: E=E-2 :: P=P-1 :: X=S1 :: Y=S2
2100 DISPLAY AT(24,1):"Trajectoire torp.:
"
2110 X=X+X1 :: Y=Y+X2 :: X3=INT(X+.5):: Y
3=INT(Y+.5)
```

```
2120 IF X3<1 OR X3>8 OR Y3<1 OR Y3>8 THEN
 2310
2130 DISPLAY AT(24,20)BEEP:X3;",";Y3 :: C
ALL DELAI :: CALL TEST(X3,Y3,Q$," ",Z3)
:: IF Z3<>0 THEN 2110
2140 CALL TEST(X3,Y3,Q$,"K",Z3):: IF Z3=0
 THEN 2190
2150 TXT$="*** KLINGON d(truit ***" :: CA
LL BANDE(TXT$,VIT):: K3=K3-1 :: K9=K9-1
 :: IF K9<=0 THEN 2800
2160 FOR I=1 TO 3 :: IF X3=K(I,1)AND Y3=K
(I,2)THEN 2180
2170 NEXT I :: I=3
2180 K(I,3)=0 :: GOTO 2290
2190 CALL TEST(X3,Y3,Q$,"*",Z3):: IF Z3=0
 THEN 2220

2200 TXT$="L'(toile en "&STR$(X3)&" , "&S
TR$(Y3)&" a annul( l'(nergie de la torp
ille ..." :: CALL BANDE(TXT$,VIT)
2210 GOSUB 2650 :: GOTO 870
2220 CALL TEST(X3,Y3,Q$,"B",Z3):: IF Z3=0
 THEN 2060
2230 TXT$="*** STARBASE d(truite ***" ::
CALL BANDE(TXT$,VIT):: B3=B3-1 :: B9=B9
-1
2240 IF B9>0 OR K9>T-T0-T9 THEN 2270
2250 TXT$="Ca suffira !! Vous etes relev(
 de votre commandement et condamn( @ 99
 starjours de travaux forc(s "
2260 TXT$=TXT$&"sur CYGNUS 12 ..." :: CAL
L BANDE(TXT$,VIT):: GOTO 2870
2270 TXT$="Commandement Flotte Stellaire
revoit votre dossier et envisage cour m
artiale ..."
2280 CALL BANDE(TXT$,VIT):: D0=0
2290 CALL SETSECT(X3,Y3,Q$," ")
2300 Z(Q1,Q2),G(Q1,Q2)=K3*100+B3*10+S3 ::
 GOSUB 2650 :: CALL COND(K3,S):: GOTO 8
50
2310 TXT$="La torpille a manqu( le but ..
." :: CALL BANDE(TXT$,VIT):: GOSUB 2650
 :: GOTO 870
2320 !================
2330 !=energie ecrans=
2340 !================
2350 IF D(7)<0 THEN TXT$="`CONTROLE ECRAN
S' ***hors service***" :: CALL BANDE(TX
T$,VIT):: GOTO 870
2360 DISPLAY AT(23,1):"Unit(s pour les (c
rans:":"Energie disponible:";E+S
2370 ACCEPT AT(23,25)VALIDATE(DIGIT)SIZE(
4)BEEP:X :: IF S=X THEN TXT$="" :: GOTO
 2400
2380 IF X<=S+E THEN 2410
2390 TXT$="Energie disponible: "&STR$(S+E
)&" unit(s. "
2400 TXT$=TXT$&"Ecrans sans changement ..
." :: CALL BANDE(TXT$,VIT):: GOTO 870
2410 E=E+S-X :: S=X :: TXT$="`CONTROLE DE
FLECTEURS' rend compte: Ecrans mis @ "&
STR$(INT(S))&" unit(s"
2420 TXT$=TXT$&" sur votre ordre ..." ::
CALL BANDE(TXT$,VIT):: CALL COND(K3,S):
: GOTO 850
2430 !==================
2440 !=rapport d'avaries=
2450 !==================
2460 IF D(6)>=0 THEN 2570
2470 TXT$="Impossible d'obtenir rapports
d'avaries ..." :: CALL BANDE(TXT$,VIT):
: IF D0=0 THEN 870
2480 D3=0 :: FOR I=1 TO 8 :: IF D(I)<0 TH
EN D3=D3+.1
2490 NEXT I :: IF D3=0 THEN 870
2500 D3=D3+D4 :: IF D3>=1 THEN D3=.9
2510 TXT$="Techniciens prets @ r(parer vo
tre Vaisseau ..." :: CALL BANDE(TXT$,VI
T)
2520 TXT$="Temps de r(paration estim( @ "
&STR$(INT(10*D3+5)/10)&" starjours ..."
 :: CALL BANDE(TXT$,VIT)
2530 DISPLAY AT(23,1):"Autorisation r(par
ation?" :: ACCEPT AT(23,26)VALIDATE("ON
")SIZE(1)BEEP:A$
2540 IF A$<>"O" THEN 870
2550 FOR I=1 TO 8 :: IF D(I)<0 THEN D(I)=
0
2560 NEXT I :: T=T+D3+.1
2570 SRS=0 :: CALL RAZ(LIG,LIG+11)
2580 DISPLAY AT(LIG,24):"AVANC":"ELEMENT"
;TAB(24);"REPAR":RPT$("_",28):: FOR I=1
 TO 8
2590 CALL MATER(I,G$):: DISPLAY AT(LIG+2+
I,1):G$;TAB(24);INT(D(I)*100)/100
2600 NEXT I :: DISPLAY AT(LIG+11,1):RPT$(
"_",28):: IF D0<>0 THEN 2480
2610 GOTO 870
2620 !=================
2630 !=tir des klingons=
2640 !=================
2650 IF K3<=0 THEN RETURN
2660 IF D0<>0 THEN TXT$="ENTERPRISE prot(
g(e par (crans de la STARBASE ..." :: C
```

```
ALL BANDE(TXT$,VIT):: RETURN
2670 FOR I=1 TO 3 :: IF K(I,3)<=0 THEN 27
50
2680 H=INT(K(I,3)/(SQR((K(I,1)-S1)^2+(K(I
,2)-S2)^2))*(2+RND)):: S=S-H :: K(I,3)=
K(I,3)/(3+RND)
2690 TXT$=STR$(H)&" unit(s frappent l'ENT
ERPRISE venant du secteur "&STR$(K(I,1)
)&" , "&STR$(K(I,2))&" ..."
2700 CALL BANDE(TXT$,VIT):: IF S<=0 THEN
2840
2710 TXT$="Ecrans tomb(s @ "&STR$(S)&" un
it(s ..." :: CALL BANDE(TXT$,VIT):: IF
H<20 THEN 2750
2720 IF RND>.6 OR H/S<=.02 THEN 2750
2730 R1=INT(RND*8+1):: D(R1)=D(R1)-H/S-RN
D/2 :: CALL MATER(R1,G$)
2740 TXT$="`CONTROLE AVARIES' signale: "&
G$&" endommag( par le coup ..." :: CALL
 BANDE(TXT$,VIT)
2750 NEXT I :: RETURN
2760 !==============
2770 !=fin de partie=
2780 !==============
2790 !gagne
2800 CALL CLEAR :: DISPLAY AT(1,1):"FELIC
ITATIONS, Commandant.":"Le dernier croi
seur KLINGON"
2810 DISPLAY AT(3,1):"mena[ant la F(d(rat
ion a (t(":"d(truit."
2820 DISPLAY AT(6,1):"Votre taux d'effica
cit( est":"de";INT(10000*(K7/(T-T0))^2)
/10;"." :: GOTO 2880
2830 !perdu
2840 CALL RAZ(1,4):: DISPLAY AT(1,1):"L'E
NTERPRISE a (t( d(truite.":"":"La F(d(r
ation sera conquise."
2850 !abandon (XXX)
2860 CALL RAZ(5,24):: DISPLAY AT(6,1):"C'
est le starjour ";STR$(T);"."
2870 DISPLAY AT(8,1):"Il y avait encore";
K9;"croi-":"seurs de bataille KLINGON @
":"la fin de votre mission."
2880 IF B9=0 THEN CALL DELAI :: CALL DELA
I :: GOTO 2930
2890 CALL HCHAR(12,3,128,28):: DISPLAY AT
(13,1):"La F(d(ration Galactique":"cher
che un autre Commandant"
2900 DISPLAY AT(15,1):"de vaisseau stella
ire pour":"une mission similaire.":"":"
S'il y a un volontaire, que"
2910 DISPLAY AT(19,1):"celui-ci sorte des
 rangs et":"frappe `DAC':" :: CALL HCHA
R(21,3,128,28)
2920 ACCEPT AT(20,16)VALIDATE(UALPHA)SIZE
(-3)BEEP:REP$ :: IF REP$="DAC" THEN 390
2930 CALL CLEAR :: STOP
2940 !================
2950 != detecteurs =
2960 !=courte distance=
2970 !================
2980 IF D(2)<0 THEN TXT$="SENSEURS COURTE
 DISTANCE ***hors service***" :: CALL B
ANDE(TXT$,VIT):: RETURN
2990 FOR I=S1-1 TO S1+1 :: FOR J=S2-1 TO
S2+1 :: IF I<1 OR I>8 OR J<1 OR J>8 THE
N 3010
3000 CALL TEST(I,J,Q$,"B",Z3):: IF Z3=1 T
HEN 3020
3010 NEXT J :: NEXT I :: D0=0 :: GOTO 304
0
3020 D0=1 :: C$="Entr( au Bassin" :: E=E0
 :: P=P0
3030 TXT$="Ecrans @ ZERO pour l'entr(e en
 radoub ..." :: CALL BANDE(TXT$,VIT)::
S=0 :: GOTO 3060
3040 IF K3>0 THEN C$="Rouge" ELSE IF E<E0
/10 THEN C$="Jaune" ELSE C$="Verte"
3050 C$="Condition "&C$
3060 SRS=1 :: CALL RAZ(LIG,LIG+11):: DISP
LAY AT(LIG,3):"SENSEURS COURTE DISTANCE
":RPT$("_",28)
3070 FOR I=1 TO 8 :: DISPLAY AT(LIG+1+I,1
):STR$(I);"I";SEG$(Q$,(I-1)*8+1,8);"I"
3080 ON I GOTO 3090,3100,3110,3120,3130,3
140,3150,3160
3090 DISPLAY AT(LIG+1+I,12):"Starjour";IN
T(T*10)/10 :: GOTO 3170
3100 DISPLAY AT(LIG+1+I,12):C$ :: GOTO 31
70
3110 DISPLAY AT(LIG+1+I,12):"Quadrant";Q1
;",";Q2 :: GOTO 3170
3120 DISPLAY AT(LIG+1+I,12):"Secteur ";S1
;",";S2 :: GOTO 3170
3130 DISPLAY AT(LIG+1+I,12):"Torp. phot.
";P :: GOTO 3170
3140 DISPLAY AT(LIG+1+I,12):"Energie tot.
";INT(E+S):: GOTO 3170
3150 DISPLAY AT(LIG+1+I,12):"dont (crans
";INT(S):: GOTO 3170
3160 DISPLAY AT(LIG+1+I,13):"=";K9;"KLING
ONS ="
```

```
3170 NEXT I :: DISPLAY AT(LIG+10,1):RPT$(
"_",28):" I12345678I" :: RETURN
3180 !===========
3190 !=librairie =
3200 !=ordinateur=
3210 !===========
3220 IF D(8)<0 THEN TXT$="ORDINATEUR ***h
ors service***" :: CALL BANDE(TXT$,VIT)
:: GOTO 870
3230 DISPLAY AT(23,1):"COMMANDE ORDINATEU
R:"
3240 ACCEPT AT(23,22)VALIDATE(DIGIT)SIZE(
1)BEEP:A :: IF A>6 THEN 870
3250 SRS=0 :: CALL RAZ(LIG,LIG+11):: ON A
+1 GOTO 3270,3350,3440,3530,3670,3580,3
330
3260 !commandes ordinateur (COM-0)
3270 DISPLAY AT(LIG,4):"FONCTIONS DISPONI
BLES":RPT$("_",28)
3280 DISPLAY AT(LIG+2,1):"0= Commandes Or
dinateur":"1= Enreg. galactique cumul{"
3290 DISPLAY AT(LIG+4,1):"2= Rapport de s
ituation":"3= Donn{es pour torp. phot."
3300 DISPLAY AT(LIG+6,1):"4= Donn{es Nav.
 -> Starbase":"5= Calcul Direction-Dist
ance"
3310 DISPLAY AT(LIG+8,1):"6= Carte rensei
gn{e Galaxie":RPT$("_",28):: GOTO 3230
3320 !la galaxie (COM-6)
3330 H8=0 :: GOTO 3360
3340 !carte renseignee de la galaxie (COM
-1)
3350 H8=1
3360 DISPLAY AT(LIG,4):"1  2  3  4  5  6
 7  8":TAB(3);RPT$("_",24)
3370 FOR I=1 TO 8 :: DISPLAY AT(LIG+1+I,1
):STR$(I);"I" :: IF H8=0 THEN 3400
3380 FOR J=1 TO 8 :: IF Z(I,J)=0 THEN A$=
"---" ELSE A$=SEG$(STR$(Z(I,J)+1000),2,
3)
3390 DISPLAY AT(LIG+1+I,J*3):A$ :: NEXT J
 :: GOTO 3420
3400 CALL SITU(I,1,G$):: G$=SEG$(G$,1,POS
(G$," ",1)-1):: DISPLAY AT(LIG+1+I,INT(
9-(LEN(G$)-1)/2)):G$;TAB(14);"II"
3410 CALL SITU(I,5,G$):: G$=SEG$(G$,1,POS
(G$," ",1)-1):: DISPLAY AT(LIG+1+I,INT(
21-(LEN(G$)-1)/2)):G$
3420 DISPLAY AT(LIG+1+I,27):"I" :: NEXT I
 :: DISPLAY AT(LIG+10,3):RPT$("_",24)::
 GOTO 870
3430 !rapport de situation (COM-2)
3440 DISPLAY AT(LIG,5):"RAPPORT DE SITUAT
ION":TAB(5);RPT$("_",20)
3450 IF K9>1 THEN X$="S" ELSE X$=""
3460 DISPLAY AT(LIG+3,1):"Il reste";K9;"K
LINGON";X$;" et";INT((T0+T9-T)*10)/10:"
Starjours pour accomplir vo-tre mission."
3470 IF B9<1 THEN 3490 ELSE IF B9=1 THEN
X$="" ELSE X$="s"
3480 DISPLAY AT(LIG+7,1):"La F{d{ration m
aintient";B9:"Starbase";X$;" dans la Ga
laxie." :: GOTO 3500
3490 DISPLAY AT(LIG+7,1):"Votre b{tise vo
us a livr{ @ vous-meme dans la Galaxie.
":" ":"Vous n'avez plus de Starbases!"
3500 DISPLAY AT(LIG+11,1):RPT$("_",28)
3510 TXT$="Tapez une touche pour la suite
 ..." :: CALL BANDE(TXT$,VIT):: GOTO 24
60
3520 !donnees pour torpilles photoniques
(COM-4)
3530 IF K3=0 THEN 1810 ELSE IF K3=1 THEN
X$="" ELSE X$="S"
3540 DISPLAY AT(LIG,2):"ENTERPRISE vers K
LINGON";X$:" ";RPT$("_",23+LEN(X$)):: X
=LIG+2
3550 FOR I=1 TO 3 :: IF K(I,3)>0 THEN X=X
+1 :: CALL DIRDIS(S1,S2,K(I,1),K(I,2),X
)
3560 NEXT I :: DISPLAY AT(X+1,1):RPT$("_"
,28):: GOTO 870
3570 !calculs de direction/distance (COM-
5)
3580 DISPLAY AT(LIG,1):"CALCUL DE DIRECTI
ON-DISTANCE":RPT$("_",28):"Vous etes da
ns le secteur"
3590 DISPLAY AT(LIG+3,1):STR$(S1);",";STR
$(S2);" du quadrant ";STR$(Q1);",";STR$
(Q2);"."
3600 DISPLAY AT(LIG+4,1):"Entrez:":" Quad
rant d'arriv{e (XY):":" Secteur d'arriv
{e  (XY):"
3610 ACCEPT AT(LIG+5,27)VALIDATE(DIGIT)SI
ZE(2)BEEP:REP
3620 C1=INT(REP/10):: A=REP-C1*10 :: IF C
1<1 OR C1>8 OR A<1 OR A>8 THEN 3610
3630 ACCEPT AT(LIG+6,27)VALIDATE(DIGIT)SI
ZE(2)BEEP:REP
3640 W1=INT(REP/10):: X=REP-W1*10 :: IF W
1<1 OR W1>8 OR X<1 OR X>8 THEN 3630
3650 CALL DIRDIS(Q1*8+S1,Q2*8+S2,C1*8+W1,
```

```
A*8+X,LIG+8):: DISPLAY AT(LIG+9,1):RPT$
("_",28):: GOTO 870
3660 !donnees vers starbase (COM-3)
3670 IF B3<>0 THEN 3690
3680 TXT$="SENSEURS ne montrent aucune St
arbase dans ce quadrant ..." :: CALL BA
NDE(TXT$,VIT):: GOTO 870
3690 DISPLAY AT(LIG,1):"ENTERPRISE vers S
TARBASE:",RPT$("_",25):: CALL DIRDIS(S1
,S2,B4,B5,LIG+4)
3700 DISPLAY AT(LIG+5,1):RPT$("_",28):: G
OTO 870
3710 !====================
3720 !====================
3730 !====================
3740 !recherche d'un secteur libre
3750 SUB ALLOC(I,J,Q$)
3760 I=INT(RND*8+1):: J=INT(RND*8+1):: CA
LL TEST(I,J,Q$," ",K):: IF K=0 THEN 376
0
3770 SUBEND
3780 !====================
3790 !entree dans un secteur
3800 SUB SETSECT(I,J,Q$,A$)
3810 K=(I-1)*8+J :: IF K=1 THEN Q$=A$&SEG
$(Q$,2,LEN(Q$)-K):: SUBEXIT
3820 IF K=64 THEN Q$=SEG$(Q$,1,63)&A$ ::
SUBEXIT
3830 Q$=SEG$(Q$,1,K-1)&A$&SEG$(Q$,K+1,LEN
(Q$)-K):: SUBEND
3840 !====================
3850 !test d'un secteur
3860 SUB TEST(I,J,Q$,A$,K)
3870 K=(I-1)*8+J :: IF SEG$(Q$,K,1)<>A$ T
HEN K=0 ELSE K=1
3880 SUBEND
3890 !====================
3900 !affichage direction-distance
3910 SUB DIRDIS(XS,YS,XD,YD,K)
3920 DIST=INT(SQR((XD-XS)^2+(YD-YS)^2)*10
/8+.5)/10 :: IF YS<>YD THEN 3960
3930 IF XS=XD THEN DIR=0 :: GOTO 3970
3940 IF XD<XS THEN DIR=90 ELSE DIR=270
3950 GOTO 3970
3960 DIR=-INT(ATN((XD-XS)/(YD-YS))*180/PI
+.5):: IF YD<YS THEN DIR=DIR+180 ELSE I
F XD>XS THEN DIR=DIR+360
3970 DISPLAY AT(K,1):"<Dir. =";DIR;">-<WA
RP. =";DIST;">" :: SUBEND
3980 !====================
3990 !situation galactique
4000 SUB SITU(I,J,I$)
4010 ON I+8*INT((J-1)/4)GOTO 4020,4030,40
40,4050,4060,4070,4080,4090,4100,4110,4
120,4130,4140,4150,4160,4170
4020 I$="ANTARES" :: GOTO 4180
4030 I$="RIGEL" :: GOTO 4180
4040 I$="PROCYON" :: GOTO 4180
4050 I$="VEGA" :: GOTO 4180
4060 I$="CANOPUS" :: GOTO 4180
4070 I$="ALTAIR" :: GOTO 4180
4080 I$="SAGITTAIRE" :: GOTO 4180
4090 I$="POLLUX" :: GOTO 4180
4100 I$="SIRIUS" :: GOTO 4180
4110 I$="DENEB" :: GOTO 4180
4120 I$="CAPELLA" :: GOTO 4180
4130 I$="BETELGEUSE" :: GOTO 4180
4140 I$="ALDEBARAN" :: GOTO 4180
4150 I$="REGULUS" :: GOTO 4180
4160 I$="ARCTURUS" :: GOTO 4180
4170 I$="L'EPI"
4180 ON J GOTO 4190,4200,4210,4220,4190,4
200,4210,4220
4190 I$=I$&" I" :: SUBEXIT
4200 I$=I$&" II" :: SUBEXIT
4210 I$=I$&" III" :: SUBEXIT
4220 I$=I$&" IV" :: SUBEND
4230 !====================
4240 !emission message
4250 SUB BANDE(TXT$,VIT)
4260 TXT$=RPT$(" ",28)&TXT$
4270 FOR I=1 TO LEN(TXT$):: DISPLAY AT(24
,1):SEG$(TXT$,I,28):: CALL SOUND(VIT,40
000,30)
4280 CALL KEY(0,K,S):: IF S<>0 THEN CALL
RAZ(24,24):: SUBEXIT
4290 NEXT I :: GOTO 4270 :: SUBEND
4300 !====================
4310 !effacement ecran partiel
4320 SUB RAZ(I,J)
4330 CALL HCHAR(I,1,32,(J-I+1)*32):: SUBE
ND
4340 !====================
4350 !calcul element materiel
4360 SUB MATER(I,I$)
4370 ON I GOTO 4380,4390,4400,4410,4420,4
430,4440,4450
4380 I$="MOTEURS WARP" :: SUBEXIT
4390 I$="SENSEURS COURTE DIST." :: SUBEXI
T
4400 I$="SENSEURS LONGUE DIST." :: SUBEXI
T
```

```
4410 I$="CONTROLE PHASERS" :: SUBEXIT
4420 I$="TUBES PHOTONIQUES" :: SUBEXIT
4430 I$="CONTROLE AVARIES" :: SUBEXIT
4440 I$="CONTROLE ECRANS" :: SUBEXIT
4450 I$="LIBRAIRIE-ORDINATEUR" :: SUBEND
4460 !====================
4470 !attente
4480 SUB DELAI
4490 FOR I=1 TO 1000 :: NEXT I :: SUBEND
4500 !====================
4510 !condition de combat
4520 SUB COND(I,J)
4530 CALL RAZ(6,9):: IF I=0 THEN SUBEXIT
4540 DISPLAY AT(6,7)BEEP:"ZONE DE COMBAT"
:TAB(7);"CONDITION ROUGE"
4550 IF J<200 THEN DISPLAY AT(9,2)BEEP:"E
CRANS DANGEREUSEMENT BAS"
4560 SUBEND
4570 END
```

Ecriture directe dans les registres VDP

Gérard Baroni et Jean Marin

Cet article a pour but de vous démontrer qu'il est parfaitement possible de faire des "pokes" dans les registres vidéo, en utilisant l'instruction CALL PEEKV.

Rappelons qu'il existe deux zones de mémoire vive différentes : la mémoire directement adressable par le microprocesseur (CPU) et la mémoire adressable par le processeur vidéo (VDP).

Notre intérêt se porte actuellement sur le registres internes du processeur vidéo. Ce processeur comporte 8 registres, numérotés de 0 à 7, qui décident du fonctionnement du processeur d'affichage.

Jusqu'à présent, à l'exception de *R1* dont une copie se trouve en mémoire CPU (−31788), l'accès à ces registres était impossible en BASIC. Notre propos est d'offrir un accès à ces registres du processeur vidéo depuis le Basic, avec la Mini-mémoire.

L'astuce consiste à détourner de son rôle normal l'instruction Basic CALL PEEKV (Adresse,X) et de s'en servir pour écrire ce que nous voulons dans les registres VDP. Le rôle normal de CALL PEEKV est de lire le contenu de la mémoire VDP de 0 à 16383. Nous utilisons des adresses illicites que nous calculons à l'aide de notre formule "MAGIQUE" :

$A = -32768 + 256 \star N + D$

- *N* étant le numéro du registre (de 0 à 7);
- *D* étant la valeur de l'octet de 0 à 255 que nous voulons écrire dans le registre VDP numéro *N*.

Rappelons le rôle des huit registres:

- RO = 0 - normal
- RO = 2 - bit map
- R1 = 160 - transparent
- R1 = 224 - normal
- R1 = 232 - mode multicolore
- R1 = 240 - mode texte
- R1 = 225 - sprites magnifiés
- R1 = 226 - sprites double taille
- R1 = 227 - sprites magnifiés double taille
- R2 indique le début de la Table d'écran. L'adresse étant le contenu de R2 multiplié par 1024 et pouvant s'exprimer schématiquement ainsi : (R2) ⋆ 1024
- R3 indique le début de la table des couleurs qui débute en : (R3) ⋆ 64
- R4 indique le début de la table des descripteurs : (R4) ⋆ 2048
- R5 Table d'attributs des sprites : (R5) ⋆ 128
- R6 Table des formes des sprites : (R6) ⋆ 2048
- R7 Couleur de fond d'écran et, pour le mode texte uniquement, couleur des caractères.

Les six applications suivantes ne sont bien entendu pas limitatives, et illustrent simplement quelques exemples découlant du principe général.

Ecriture transparente

Combattre le "SCROLL" et ses désagréments est simple me direz-vous ! Il suffit, cela est bien connu, de rendre l'écran transparent pendant l'exécution des *PRINT*. Plusieurs solutions ont déjà été apportées à ce problème (voir "99 Magazine" numéro 4). La nôtre est très simple et courte à écrire.

Nous mettons 160 dans R1, à la place du contenu normal (224).
En application de la formule générale nous calculons A :

$A = -32352 + 256 \star 1 + 160 = -32352$

CALL PEEKV (−32352, X) donne donc la transparence et CALL PEEKV (−32288, X) permet la réapparition du texte (224 étant remis en R1).

```
100 REM  ********************
110 REM  *  Efface l'ecran  *

120 REM  ********************

130 REM
140 CALL CLEAR
150 CALL PEEKV(-32352,X)
160 PRINT "CE PROGRAMME EVITE LA
"::"VISUALISATION DU ""SCROLL"""
::::::::::::::::
::::
170 CALL PEEKV(-32288,X)
180 GOTO 180
```

Caractères de la mire et sigle Texas

Le Groupe des caractères affichés sur la mire d'introduction n'est pas le même que celui qui utilisé en Basic. Ces grandes lettres, le sigle Texas, et le sigle "Copyright" sont dans la mémoire VDP à partir de l'adresse 2048. Pour y accéder, il suffit de déplacer, comme le montre notre exemple, la table des formes avec (R4)=1.

Nous vous laissons le soin d'analyser la suite mais, vous constaterez que l'avantage évident est de nous procurer un jeu de 256 caractères distincts en plus du premier jeu.

```
10 REM *******************
20 REM *  2eme groupe de  *
30 REM *    caracteres    *
40 REM *******************
50 REM
```

```
100 CALL CLEAR
110 CALL PEEKV(-31743,X)
120 CALL PEEKV(-31986,X)
130 FOR I=896 TO 927 STEP 8
140 CALL POKEV(I,16,16,16,16,16,
16,16,16)
150 NEXT I
160 FOR I=0 TO 255
170 PRINT CHR$(I);
180 NEXT I
190 FOR I=0 TO 2
200 FOR J=0 TO 2
210 CALL POKEV(330+J+32*I,1+J+3*
I)
220 NEXT J
230 NEXT I
240 GOTO 240
```

Commutation de deux ou plusieurs images

Cela nous permet d'écrire dans une image pendant la visualisation d'une autre image. La gestion simultanée de plusieurs pages de texte ou d'image est donc possible par déplacements de la table d'écran.

```
100 REM  *******************
110 REM  *   Double page    *
120 REM  *******************
130 REM
140 CALL CLEAR
150 PRINT "   DOUBLE PAGE D'ECRA
N":"                  ":::::
:"PATIENTEZ QUEL
QUES INSTANTS.":::
160 FOR I=2048 TO 2815 STEP 8
170 CALL POKEV(I,128,128,128,128
,128,128,128,128)
180 NEXT I
190 A$="= PREMIERE PAGE ="
200 A=2278
210 GOSUB 400
220 A$="POUR VISUALISER LA DEUXI
EME"
230 A=2402
240 GOSUB 400
250 A$="PAGE TAPEZ UNE TOUCHE"
260 A=2468
270 GOSUB 400
280 CALL PEEKV(-32254,X)
290 CALL CLEAR
300 PRINT "   *** DEUXIEME PAGE
***"::::"  EN CE MOMENT LA DEUXI
EME"::"  PAGE ES
T AFFICHEE SUR"::
310 PRINT " L'ECRAN DU TELEVISE
UR"::::"   IL EST POSSIBLE DE"::
" REVENIR TRES R
APIDEMENT"::
320 PRINT "A LA PREMIERE PAGE EN
 TAPANT"::"N'IMPORTE QUELLE TOUC
HE."::::
330 CALL PEEKV(-32254,X)
340 CALL KEY(5,R,E)
350 IF E=0 THEN 340
360 CALL PEEKV(-32256,X)
370 CALL KEY(5,R,E)
380 IF E=0 THEN 370
390 GOTO 330
400 FOR I=1 TO LEN(A$)
410 CALL POKEV(A+I,ASC(SEG$(A$,I
,1))+96)
420 NEXT I
430 RETURN
```

Mode Texte

L'accès à ce mode et à ses 40 colonnes par le Basic et la Mini-mémoire offre de bonnes possibilités d'applications, ne serait-ce qu'en conjuguant ses avantages avec ceux proposés dans l'exemple précédent.

```
100 REM  *******************
110 REM  * Mode multicolore *
120 REM  *******************
130 REM
140 CALL CLEAR
150 PRINT "     MODE MULTICOLORE
"::::"LES COMMANDES SONT:":::" C
 CHANGER LA COUL
EUR"::" D DROITE"::" S GAUCHE"::
160 PRINT " E HAUT"::" X BAS"::"
 A ALEATOIRE"::" (ENTER) FINIR":
:::
170 INPUT "(ENTER) PATIENTEZ..."
:R$
180 CALL CLEAR
190 RANDOMIZE
200 CALL PEEKV(-32280,X)
210 CALL PEEKV(-31743,X)
220 CALL LOAD(-31788,232)
230 FOR N=0 TO 5
240 FOR I=0 TO 31 STEP 8
250 D=I+32*N
260 CALL POKEV(128*N+I,D,D+1,D+2
,D+3,D+4,D+5,D+6,D+7)
270 CALL POKEV(128*N+I+32,D,D+1,
D+2,D+3,D+4,D+5,D+6,D+7)
280 CALL POKEV(128*N+I+64,D,D+1,
D+2,D+3,D+4,D+5,D+6,D+7)
290 CALL POKEV(128*N+I+96,D,D+1,
D+2,D+3,D+4,D+5,D+6,D+7)
300 CALL SOUND(100,440,0)
310 NEXT I
320 NEXT N
330 FOR A=2048 TO 3583 STEP 16
340 CALL POKEV(A,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0)
350 NEXT A
360 Y=24
370 X=32
380 C=7
390 N=X-2*INT(X/2)
400 A=2048+8*INT(X/2)+Y+248*INT(
Y/8)
410 CALL PEEKV(A,D)
420 D=C-15*C*(N=0)-D*(N=0)+16*IN
T(D/16)*((N=0)-(N=1))
430 CALL POKEV(A,D)
440 CALL SOUND(100,220,0)
450 CALL KEY(5,R,E)
460 IF E=0 THEN 450
470 IF R<>13 THEN 520
480 CALL PEEKV(-32288,X)
490 CALL LOAD(-31788,224)
500 CALL CLEAR
510 END
520 IF R<>67 THEN 550
530 C=C+1+15*(C=15)
540 GOTO 420
550 IF R<>68 THEN 580
560 X=X+1+(X=63)
570 GOTO 390
580 IF R<>83 THEN 610
590 X=X-1-(X=0)
600 GOTO 390
610 IF R<>69 THEN 640
620 Y=Y-1-(Y=0)
630 GOTO 390
640 IF R<>88 THEN 670
650 Y=Y+1+(Y=47)
660 GOTO 390
670 IF R<>65 THEN 450
680 X=INT(64*RND)
690 Y=INT(48*RND)
```

```
700 CALL SOUND(100,110+1000*RND,
0)
710 GOTO 390
```

Mode Multicolore

Ce mode particulier permet des graphismes colorés en quarts de caractère. Quatre couleurs sont accessibles par caractère au lieu de 2 en mode normal. Le programme de dessin est une petite récompense à votre assiduité.

```
100 REM *******************
110 REM *   Mode texte    *
120 REM *******************
130 REM
140 CALL CLEAR
150 A$="** AFFICHAGE EN MODE TEX
TE 40 COLONNES *"
160 FOR I=959 TO 768 STEP -1
170 CALL POKEV(I,128)
180 NEXT I
190 CALL PEEKV(-32272,X)
200 CALL PEEKV(-30957,X)
210 FOR I=1 TO LEN(A$)
220 CALL POKEV(I-1,ASC(SEG$(A$,I
,1))+96)
230 NEXT I
240 GOTO 240
```

Les Sprites

Cette pluie de 26 lettres vous démontre, si besoin était, que les lutins peuvent être totalement gérés par le Basic et la Mini-mémoire, et cela sans passer par des routines en Assembleur.

Mentionnons également que le BIT-MAP est possible mais interfère vite avec le Basic du fait d'une forte occupation de la mémoire.

```
100 REM *******************
110 REM * Pluie de lettres *
120 REM *******************
130 REM
140 RANDOMIZE
150 CALL CLEAR
160 CALL POKEV(911,16,16,16,16,1
6,16,16,16,16,16,16,16,16)
170 CALL SCREEN(1)
180 CALL PEEKV(-31986,X,"",-3228
7,Y)
190 CALL LOAD(-31788,225)
200 FOR I=2 TO 26
210 CALL POKEV(764+4*I,6*I,8*I,1
60+I,2+INT(I/2))
220 NEXT I
230 CALL POKEV(872,208,"",768,6,
8,161,2)
240 FOR I=1 TO 26
250 CALL POKEV(1916+4*I,60*RND,0
)
260 NEXT I
270 CALL LOAD(-31878,26)
280 CALL KEY(5,VR,VE)
290 IF VE=0 THEN 280
300 CALL LOAD(-31878,0)
310 CALL KEY(5,VR,VE)
320 IF VE=0 THEN 310
330 GOTO 270
```

En conclusion : la connaissance du contenu des huit registres du processeur vidéo *TMS 9929*, notre formule "magique", quelques déplacements judicieux de tables et l'utilisation d'une façon simple et ô! combien confortable par rapport au langage Assembleur, du Basic, nous permet d'accéder directement au processeur vidéo, simplifiant l'écriture et l'exécution des programmes et sans pour autant faire appel à la RAM de la Mini-mémoire qui reste libre pour d'autres usages. ■

My tailor was rich

T. Gallier

Etant professeur d'anglais, j'ai utilisé mes compétences linguistiques pour réaliser ce programme écrit **en Basic TI**. Il s'agit d'un jeu basé sur les verbes irréguliers au prétérit (qu'il faut apprendre par coeur et souvent réciter) sous la forme question / réponse.

La liste des verbes est aussi utilisé pour une seconde option baptisé "Hangman" (pendu) où l'on cherche l'infinitif d'un verbe irrégulier, puis le prétérit est affiché.

Dans un but didactique, les instructions d'utilisation sont en anglais, à savoir :

TO STOP TYPE : END (pour arrêter entrez END);
USE BLOCK CAPITALS (utilisez des majuscules);
TRY AGAIN (essayez encore);
WELL DONE (chapeau !).

Les utilisateurs intéressés par la programmation pourront chercher à perfectionner ce programme au gré de leur imagination, par exemple :

- dans la première partie (prétérit), classement des DATA pour permettre des niveaux de difficulté avec restrictions au moment du choix de la zone-matrice;
- dans la seconde partie (hangman), une fois le verbe trouvé par le joueur, retour à la première partie du programme et l'utilisateur devra trouver le prétérit;
- possibilité de retour à la première partie à l'issue de la seconde;
- affichage du score pour "le pendu", éventuellement en prenant en compte le nombre d'essais;
- amélioration du graphisme pour le pendu. ■

```
100 REM  ******************
110 REM  *    MY TAILOR   *
120 REM  *       WAS      *
130 REM  *      RICH      *
140 REM  ******************
150 REM  * revision du passe*
160 REM  * des verbes irre- *
170 REM  * guliers anglais  *
180 REM  ******************
190 REM  *    BASIC TI    *
200 REM  ******************
210 REM
```

```
220 CALL CLEAR
230 CALL SCREEN(10)
240 X=850
250 PRINT TAB(10);"DO YOU WANT"::" THE PR
ETERIT OF IRREGULAR      VERBS (1)"::
260 PRINT ::
270 PRINT TAB(10);"OR"::
280 PRINT TAB(2);"THE HANGMAN (2) ?"
290 INPUT C
300 CALL CLEAR
310 IF C=2 THEN 1450
320 PRINT TAB(9);"PRETERIT  OF"::TAB(8);"
IRREGULAR VERBS":"    *****************
***"::
330 GOTO 870
340 PRINT TAB(6);"TO STOP TYPE: END"
350 PRINT ::
360 PRINT ::
370 PRINT TAB(15);"(c) T.Gallier"
380 OPTION BASE 1
390 DIM V$(6,2,11)
400 FOR L=1 TO 6
410 FOR J=1 TO 2
420 FOR I=1 TO 11
430 READ V$(L,J,I)
440 NEXT I
450 NEXT J
460 NEXT L
470 DATA "GO","EAT","SIT","GET","TAKE","G
IVE","RUN","FLY","BUY","THINK","BREAK"
480 DATA "WENT","ATE","SAT","GOT","TOOK",
"GAVE","RAN","FLEW","BOUGHT","THOUGHT",
"BROKE"
490 DATA "COME","DO","KNOW","BITE","WEAR"
,"HAVE","FIND","LOSE","THROW","BEGIN","
LEARN"
500 DATA "CAME","DID","KNEW","BIT","WORE"
,"HAD","FOUND","LOST","THREW","BEGAN","
LEARNT"
510 DATA "BUILD","HEAR","READ","STAND","C
UT","DRIVE","PUT","SPEAK","SEND","BRING
","FIGHT"
520 DATA "BUILT","HEARD","READ","STOOD","
CUT","DROVE","PUT","SPOKE","SENT","BROU
GHT","FOUGHT"
530 DATA "LIGHT","MAKE","SAY","PAY","WAKE
","WIN","LEAVE","STEAL","COST","DRINK",
"FREEZE"
540 DATA "LIT","MADE","SAID","PAID","WOKE
","WON","LEFT","STOLE","COST","DRANK","
FROZE"
550 DATA "HIDE","WRITE","SHUT","SPEND","L
ET","RING","BEAT","TEACH","LEND","SWIM"
,"HIT"
560 DATA "HID","WROTE","SHUT","SPENT","LE
T","RANG","BEAT","TAUGHT","LENT","SWAM"
,"HIT"
570 DATA "SEE","SLEEP","TELL","CATCH","FE
EL","SHOOT","SELL","KEEP","FALL","MEET"
,"CHOOSE"
580 DATA "SAW","SLEPT","TOLD","CAUGHT","F
ELT","SHOT","SOLD","KEPT","FELL","MET",
"CHOSE"
590 CALL CLEAR
600 IF OPT$="YES" THEN 700
610 PRINT "WHAT'S YOUR NAME?"
620 INPUT NAME$
630 PRINT ::
640 PRINT "HELLO ";NAME$;" !"
650 PRINT "  NOW LET'S START..."::
660 FOR DELAI=1 TO 300
670 NEXT DELAI
680 T=0
690 S=0
700 RANDOMIZE
710 IF OPT$="YES" THEN 730
720 IF C=1 THEN 750
730 P=INT(RND*5)+1
740 GOTO 760
750 P=INT(RND*6)+1
760 K=INT(RND*11)+1
770 IF C=2 THEN 1560
780 IF OPT$="YES" THEN 1560
790 IF T<2 THEN 810
800 CALL CLEAR
810 PRINT "PRETERIT OF"::"TO",V$(P,1,K)
820 X=-500
830 GOTO 1280
840 IF R$="END" THEN 1050
850 IF R$=V$(P,2,K)THEN 860 ELSE 980
860 PRINT "CORRECT !!"::
870 CALL SOUND(X,698,2)
880 CALL SOUND(X,698,2)
890 CALL SOUND(X,784,2)
900 CALL SOUND(X,659,2)
910 CALL SOUND(X,698,2)
920 CALL SOUND(X,784,2)
930 IF C=2 THEN 1540
940 IF OPT$="YES" THEN 1540
950 IF T=0 THEN 340
960 S=S+1
970 IF R$<>"END" THEN 750 ELSE 1140
```

```
980 CALL SOUND(2000,-4,5)
990 PRINT TAB(10);"***";V$(P,2,K);"***":
1000 PRINT TAB(11);"(";"TO ";V$(P,1,K);")
"::
1010 FOR DELAI=1 TO 1500
1020 NEXT DELAI
1030 CALL CLEAR
1040 GOTO 760
1050 PRINT ::
1060 PRINT NAME$;" SCORES :"::
1070 PRINT TAB(5);S,"OUT OF ";T-1::
1080 X=500
1090 TM=INT(T*2/3)
1100 IF S>=TM THEN 870
1110 FOR I=1 TO 8
1120 CALL SOUND(1000,-I,3)
1130 NEXT I
1140 CALL CLEAR
1150 PRINT "DO YOU WANT TO PLAY AGAIN ";N
AME$;"?"::
1160 PRINT "TYPE 'YES' OR 'NO'"::
1170 INPUT CONT$
1180 IF CONT$="YES" THEN 650
1190 PRINT "IS THERE ANOTHER PLAYER?"::"
(TYPE 'YES' OR 'NO')"::
1200 INPUT OP$
1210 IF OP$="YES" THEN 590
1220 PRINT "DO YOU WANT TO PLAY ""THE HAN
GMAN""?"
1230 INPUT OPT$
1240 IF OPT$="YES" THEN 1440
1250 PRINT "BYE  ";NAME$;"! ";" SEE YOU A
GAIN..."
1260 END
1270 CALL SOUND(1000,-4,3)
1280 FOR I=1 TO 32
1290 CALL SCREEN(12)
1300 CALL COLOR(4,2,7)
1310 CALL HCHAR(1,1,62,I)
1320 FOR DELAI=1 TO 100
1330 NEXT DELAI
1340 CALL KEY(3,A,ST)
1350 IF ST=0 THEN 1410
1360 A$=CHR$(A)
1370 INPUT A$:R$
1380 T=T+1
1390 R$=A$&R$
1400 GOTO 840
1410 NEXT I
1420 IF I=32 THEN 980
1430 GOTO 840
1440 CALL CLEAR
1450 PRINT TAB(10);"THE HANGMAN"::
1460 PRINT TAB(10);"***********"::
1470 PRINT ::
1480 PRINT TAB(5);"10 POSSIBLE MISTAKES":
:
1490 PRINT TAB(9);"!!!!!!!!!!"::
1500 PRINT TAB(5);"TO STOP TYPE:""/"""::
1510 PRINT TAB(5);"USE BLOCK CAPITALS"
1520 X=1200
1530 GOTO 870
1540 RESTORE
1550 GOTO 380
1560 M$=V$(P,1,K)
1570 L=LEN(M$)
1580 CALL CLEAR
1590 T$="--------"
1600 T1$=SEG$(T$,1,L)
1610 PRINT TAB(13);T1$
1620 FOR DELAI=1 TO 500
1630 NEXT DELAI
1640 SC=0
1650 INPUT A$
1660 IF A$="/" THEN 1880
1670 R=ASC(A$)
1680 FOR X=1 TO L
1690 IF SEG$(M$,X,1)=A$ THEN 1760
1700 NEXT X
1710 SC=SC+1
1720 CALL HCHAR(1,10,33,SC)
1730 CALL SOUND(750,-8,4)
1740 IF SC>10 THEN 1830
1750 GOTO 1650
1760 T1$=SEG$(T1$,1,X-1)&A$&SEG$(T1$,X+1,
L)
1770 PRINT "CORRECT"
1780 PRINT TAB(13);T1$
1790 IF T1$=M$ THEN 1800 ELSE 1650
1800 PRINT TAB(13);V$(P,2,K)
1810 PRINT "WELL DONE ";NAME$;"!!!"::
1820 GOTO 1850
1830 PRINT TAB(13);M$
1840 CALL HCHAR(24,15,42,L)
1850 FOR DELAI=1 TO 500
1860 NEXT DELAI
1870 GOTO 700
1880 PRINT "GOODBYE ";NAME$;" !!!"
1890 END
```

Le code de la route

Jacques Keck

Ce jeu écrit en **Basic Etendu** vous pose cinquante questions relatives à la signalisation routière.

Il vous faudra reconnaître les panneaux affichés, et donner la solution exacte parmi les cinq définitions que vous propose l'ordinateur. Facile... pas si sûr... car si l'ordinateur peut choisir plusieurs fois le même panneau parmi les 65 panneaux différents stockés dans sa mémoire, il n'accepte pas que fassiez plus de cinq erreurs.

Ce logiciel est, à notre avis, un excellent outil pédagogique pour apprendre à nos enfants la signalisation routière.

■

```
100 ! *********************
110 ! *  CODE DE LA ROUTE  *
120 ! *********************
130 ! *    BASIC ETENDU    *
140 ! *********************
150 !
160 DATA virage a gauche,000000000F3F0F01
000000000000000000000000080C0E0F078383
838382828
170 DATA virage a droite,00000000000010307
0F1E1C1C1C1C141400000000F0FCF0800000000
000000000
180 DATA virages dont `er a droite,000000
0000001C3E3F3B3B39383838280404OE0E0E0E0
E0E0E8ECEFC78300000
190 DATA virages dont `er a gauche,202070
70707070707071737F3E1C0000000000000003
87CFCDC9C1C1C1C1C14
200 DATA chaussee glissante,0000010301010
20C101009090408106000209088C4E274383C48
8000C0300810
210 DATA dos d ane,00000000000000000000018
3C7EFFFF0000000000000000000000183C7EFFF
F00
220 DATA pont mobile,000000000103060C3810
60E0F0F9FF00002070C08000000000000006070F9
FFF00
230 DATA danger particulier,0001030303030
3030101010100010301008OC0C0C0C0C08080
80800080C080
240 DATA chaussee retrecie,000E0E0E1C3870
E0E0E0E0E0E0E0E00000707070381C0E0707070
70707070700
250 DATA chaussee retrecie a gauche,000E0
E0E1C3870E0E0E0E0E0E0E000001C1C1C1C1C
1C1C1C1C1C1C1C1C1C00
260 DATA chaussee retrecie a droite,00383
8383838383838383838383838380000707070381C
0E070707070707070700
270 DATA circulation bilaterale,000010387
CFE38383838383838380000000001C1C1C1C1C1C
1C1C7F3E1C080000
280 DATA vent lateral,0103040A112B4C70000
0000000000000E0108C546484040404040404
040400
290 DATA attention aux enfants,0000000000
3838101C7DBABC7C24246C001C1C1C081C7E5DD
D1C7E7F14141436
300 DATA passage pour pieton,07070702070F
1767070555080819 00AA000000000000804020200
05480808000AA
310 DATA debouche de cycliste,0303070E0F0
E0E0E0F0E728ABC887000808000011F11F080888
E84E5195510E00
320 DATA aire de danger aerien,6070381F0F
0F0F0D18103020200000001C38E0C0800080C0E
0703E1810100000
330 DATA descente dangereuse,000000000004
0D05C5F1FCFFFFFFFFFF000000000000D85942C
40813C3F0FCFF
340 DATA danger animaux domestiques,00403
078FF7F1F0F0F0F0F050D0909000000000FFFD
FDFDFDFCFC14140A0A00
350 DATA passage animaux sauvages,090F083
8781C0F07070F70402000000000000000000000
C2FCF838301C020100
360 DATA pas a niveau sans barriere,00010
3070F0E08097F7F7FFF7F6DFF0000E0F0800000
3F38F8F8FEFEF0B0FF00
370 DATA pas a niveau a barrieres,006D6DF
FFF6D6D6D6D6D6DFFFF6D6D0000B6B6FFFFB6B6
B6B6B6B6FFFFB6B600
380 DATA chutes de pierres,00010300000606
0C000006644002868E80406060F0F0F8787C7C7
EFEFFFF7F7F
390 DATA priorite a droite,0000E070381C0E
07070E1C3870E000000000070E1C3870E0E0703
81C0E070000
```

```
400 DATA route prioritaire,01030707070707
3F0707070707070060480C0E0E0E0E0E0FCE0E0E
0E0E0E06020
410 DATA travaux temporaires,030303021F2F
2F1F0F0F090909094DFF808080008040404 0C02
0100C0E1EBFFF
420 DATA gravillons,0000000000001E21217F7
F7F2121FF0000000000000010002008588922050
FF00
430 DATA arret obligatoire,000000000006E
44642464000000000000000000000000EEAAAEA8E
8
440 DATA sens interdit,0000000000FFFFFFFF
FFFF0000000000000000000000FFFFFFFFFFFF000
0000000
450 DATA stationnement interdit,0F1F0F47E
3F1F8FCFEFFFFFFF7F7F7F0FF0FEFEFEFFFFFF7F
3F1F8FC7E2F0F8F0
460 DATA arret interdit,0F1F0F47E3F1F8FCF
CF8F1E3470F1F0FF0F8F0E2C78F1F3F3F1F8FC7
E2F0F8F0
470 DATA interdit dans les { sens,0F7F7F7
FFFFFFFFFFFFFFFFF7F7F7F0FF0FEFEFEFFFFFF
FFFFFFFFFFFFEFEFEF0
480 DATA obligation de tourner,0000060C18
3060FFFF6030180C06000000000000000000FEF
E000000000000
490 DATA sens giratoire,02076FE6C2C0C0C06
8383878000C0F0700C0F0381C0E060600003C1C
1C34E0C0
500 DATA vitesse minimale,00007E7E6060607
E7E0606067E7E000000007E7E66666666666666
667E7E0000
510 DATA couloir d autobus,000000000007F
5191FFFFE7180000000000000000000FF1111FFF
FE718000000
520 DATA piste cyclable,000000000E10100F0
808147299A88970000000000001C08F80810304E
D5D5110E
530 DATA voie pietonniere,003838381038 7CB
AB9B8382828286C00000000001C1C081CBE5D
1C3E141436
540 DATA "fin de ",000000000000000103070E
1C3870E0C003070E1C3870E0C080
550 DATA route prioritaire,0103070F1F3F7F
FFFF7F3F1F0F07030180C0E0F0F8FCFEFFFFFFEF
CF8F0E0C080
560 DATA cedez le passage,FFFF7F7F3F3F1F1
F0F0F07070303010 1FFFEFEFCFCF8F8F0F0E0E0
C0C08080
570 DATA parc de stationnement,0F0F0F0F0F
0F0F0F0F0F0F0F0F0F00F0F8FCFE1E1EFEFCF
8F0
580 DATA hopital faites silence,787878787
8787F7F7F7F787878787800 1E1E1E1E1E1EFEFE
FEFE1E1E1E1E1E
590 DATA "",003838383838383838383838FE7C1
0000000000000000000
600 DATA sens prioritaire,000000000000000
00000000000000000183C7E18181818181818
181818
610 DATA "",000F0F0F00000000000000000000000
00000F0F0F000000000
620 DATA voie sans issue,000000000303030 3
030303030303000000000C0C0C0C0C0C0C0C0C
0C0C0
630 DATA "",C0E070381C0E07030100000000000
0000000000000000080C0E070381C0E0703
640 DATA arret au peage,AEAAEEAAAA007FFFF
F7F00EDA9ED898DBB9293929300FEFFFFFE00DB
52D3565B
650 DATA interdit aux poids lourds,001F1F
1F7F9F9FFFFFFF9F60600000000FFFFFFFFFF
FFFFFFFF30C0C
660 DATA interdit aux pietons,03030301030
70B13630302040408281080808000 80C0A09090
90804040202060
670 DATA limitation de tonnage,00003777F6
B63636363636370000000000C0CCCECECCCCC
CCCCFC7
680 DATA largeur limitee,00003E3E06069EDE
C6863E3E000000000000000000F9ABABA9
690 DATA hauteur limitee,030100F8F81B1B7B
7B1B1BF8F8000103C080000000FFFF333333330
0000080C0
700 DATA "",000000003E22227F7F7F222200000
0000000000000000000
710 DATA Klaxon interdit,0080C0E0F0F8FCFF
FFFCF8F0E0C080000000000030F3FFFFF3F0F0
3
720 DATA ne pas tourner a gauche,2060FFFF
FF602000000000000000000000000FCFEFE0E0E0
E0E0E0E0E0E0E0E
730 DATA ne pas tourner a droite,00003F7F
7F7070707070707070707000406FFFFFF06040
000
740 DATA pas de demi tour,001F3F383030303
0FC783000000000000F0F8381818181818181818
1818181818
750 DATA interdit autos et motos,00000000
000000002040E77F4C4AA5423E22227F5D7F222
200
```

```
760 DATA interdit de depasser,00000000000
000000000000000000000000003E22227F7F
7F2222
770 DATA vitesse limitee,00007E7E6060607E
7E0606067E7E000000007E7E666666666666666
67E7E0000
780 DATA depassem interdit,000000003E2222
7F7F7F2222000000000000000003E22227F7F7F2
222
790 F=92 :: G=100 :: H=108 :: I=120 :: J=
128 :: K=132 :: N=136 :: P=140
800 V$=RPT$("0",16):: P$=RPT$("F",16):: C
ALL MAGNIFY(3):: E=0
810 CALL CHAR(96,"0000001030501010",123,"
0000003048102078")
820 F$="0103070F1F3F7FFF"
830 G$="FF7F3F1F0F070301"
840 H$="80C0E0F0F8FCFEFF"
850 I$="FFFEFCF8F0E0C080"
860 CALL CLEAR :: FOR A=2 TO 8 :: CALL CO
LOR(A,7,1):: NEXT A
870 DISPLAY AT(1,5):"LE CODE DE LA ROUTE"
880 DISPLAY AT(3,2):"QUEL EST CE PANNEAU
?" :: DISPLAY AT(5,1):"1":"2":"3":"4":"
5"
890 FOR TIRAGE=1 TO 50 :: W$=""
900 FOR A=9 TO 12 :: CALL COLOR(A,1,1)::
NEXT A
910 RANDOMIZE :: CH=INT(RND*20)+1
920 ON CH GOTO 930,1530,1170,1220,930,122
0,1530,930,1220,930,930,1220,1530,930,1
360,1430,1470,1530,1220,1470
930 Z=12 :: T=10 :: Y=INT(RND*27)+1 :: IF
 Y=PA THEN 32767 ELSE PA=Y :: L=H :: O=
J :: GOSUB 1650 :: GOSUB 1730
940 IF PA=13 THEN C=9 ELSE C=5
950 ON PA GOSUB 1840,1850,1860,1870,1880,
1890,1900,1910,1920,1930,1940,1950,1960
,1970,1980,1990,2000,2010,2020,2030,2040,
2050,2060,2070,2080,2090,2100
960 READ MT$,CA$ :: CALL CHAR(36,CA$):: G
OSUB 1760
970 V=INT(RND*5)+1 :: FOR S=1 TO 5 :: M(S
)=0 :: NEXT S :: M(V)=1 :: MOT$(V)=W$&M
T$
980 FOR S=1 TO 5
990 IF M(S)=0 THEN 1000 ELSE 1010
1000 M(S)=1 :: CALL ACX(AT$,MT$):: MOT$(S
)=AT$
1010 DISPLAY AT(4+S,3):MOT$(S)
1020 NEXT S
1030 FOR A=9 TO 12 :: CALL COLOR(A,6,1)::
 NEXT A
1040 CALL SPRITE(#2,36,C,L,O)
1050 DISPLAY AT(18,1):"VOTRE REPONSE[1,2,
3,4,5]?= 1" :: ACCEPT AT(18,28)SIZE(-1)
BEEP VALIDATE("12345"):RP
1060 IF RP=V THEN 1070 ELSE 1080
1070 DISPLAY AT(20,1):"BRAVO,LA REPONSE E
ST EXACTE" :: GOTO 1090
1080 E=E+1 :: DISPLAY AT(20,1):"FAUX,LA B
ONNE REPONSE ETAIT": :MOT$(V)
1090 FOR DE=1 TO 1000 :: NEXT DE
1100 DISPLAY AT(20,1):"": :""
1110 DISPLAY AT(24,1):"SUR";TIRAGE;"PANNE
AUX";E;"ERREURS" :: FOR DE=1 TO 100 ::
NEXT DE
1120 IF E=5 THEN CALL DELSPRITE(ALL):: GO
TO 1150 ELSE 1130
1130 CALL DELSPRITE(ALL):: NEXT TIRAGE
1140 CALL CLEAR :: DISPLAY AT(10,1):"BRAV
O VOUS AVEZ REUSSI       L'EPREUVE THEOR
IQUE DU        PERMIS DE CONDUIRE" :: END
1150 CALL CLEAR :: DISPLAY AT(10,1):"HELA
S,VOUS AVEZ ECHOUE A     L'EPREUVE THEOR
IQUE DU        PERMIS DE CONDUIRE...
REVOYEZ VOTRE CODE"
1160 DISPLAY AT(20,1):"SUR";TIRAGE;"PANNE
AUX,5 ERREURS" :: END
1170 Q$=F$&P$&P$&P$
1180 R$=P$&G$&P$&P$
1190 T$=P$&P$&H$&P$
1200 U$=P$&P$&P$&I$ :: Z=16 :: T=10
1210 GOSUB 1730 :: L=G :: O=J :: C=9 :: G
OSUB 2110 :: GOTO 960
1220 Q$="00000103070F1F3F3F7F7F7FFFFFFFFF
0F7FFFFFFFFFFFFF"&P$
1230 R$="FFFFFFFF7F7F7F3F3F1F0F0703010000
"&P$&"FFFFFFFFFFFF7F0F"
1240 T$="F0FEFFFFFFFFFFFF"&P$&"000080C0E0
F0F8FCFCFEFEFEFFFFFFFF"
1250 U$=P$&"FFFFFFFFFFFFFEF0FFFFFFFFFEFEF
EFCFCF8F0E0C0800000" :: RANDOMIZE :: R=
INT(RND*4)+1 :: ON R GOTO 1260,1290,1320,
1630 ::
1260 Z=10 :: T=16 :: GOSUB 1730 :: L=G ::
 O=J
1270 GO=INT(RND*4)+1 :: IF GO=G5 THEN 127
0 ELSE G5=GO :: IF G5=2 OR G5=3 THEN C=
5 ELSE C=16
1280 ON G5 GOSUB 2120,2130,2140,2150 :: G
OTO 960
```

```
1290 Z=5 :: T=16 :: GOSUB 1730 :: L=G ::
O=J :: C=16
1300 G2=INT(RND*6)+1 :: IF G2=G1 THEN 130
0 ELSE G1=G2
1310 ON G1 GOSUB 2160,2170,2180,2190,2200
,2210 :: GOTO 960
1320 Z=5 :: T=16 :: GOSUB 1730 :: L=G ::
O=J :: C=16
1330 G4=INT(RND*4)+1 :: IF G4=G3 THEN 130
0 ELSE G3=G4
1340 D=10 :: GOSUB 1690
1350 ON G3 GOSUB 2180,2190,2200,2210 :: G
OTO 960
1360 Q$=V$&F$&F$&P$
1370 R$=G$&V$&P$&G$
1380 T$=H$&P$&V$&H$
1390 U$=P$&I$&I$&V$ :: Z=16 :: T=5 :: CA=
INT(RND*2)+1
1400 GOSUB 1730 :: L=G :: O=J :: ON CA GO
TO 1410,1420
1410 D=2 :: GOSUB 1690
1420 RESTORE 550 :: C=11 :: GOTO 960
1430 O$="FFFF7F7F3F3F1F1F0F0F070703030101
" :: L$="FFFFFEFEFCFCF8F8F0F0E0E0C0C080
80"
1440 Q$=O$&P$&P$ :: R$=V$&V$&O$
1450 T$=P$&P$&L$ :: U$=L$&V$&V$
1460 Z=10 :: T=6 :: GOSUB 1730 :: L=G ::
O=J :: C=12 :: GOSUB 2220 :: GOTO 960
1470 Q$=P$&P$&P$&P$ :: R$=Q$ :: T$=Q$ ::
U$=Q$
1480 Z=5 :: T=16 :: GOSUB 1730 :: Q1=INT(
RND*2)+1 :: ON Q1 GOTO 1490,1500
1490 L=G :: O=J :: C=16 :: Q2=INT(RND*2)+
1 :: ON Q2 GOSUB 2230,2240 :: GOTO 960
1500 D=10 :: C=16 :: L=G :: O=J :: Q3=INT
(RND*2)+1 :: ON Q3 GOTO 1510,1520
1510 GOSUB 1700 :: GOSUB 2250 :: GOTO 960
1520 GOSUB 1710 :: GOSUB 2260 :: GOTO 960
1530 Q$="00000103070F1F3F3F7E7E7EFCFCFCFC
0F7FFFFFFFFFF080"&V$
1540 R$="FCFCFCFC7E7E7E3F3F1F0F0703010000
"&V$&"80F0FFFFFFFF7F0F"
1550 T$="F0FEFFFFFFFF0F01"&V$&"000080C0E0
F0F8FCFC7E7E7E3F3F3F3F"
1560 U$=V$&"010FFFFFFFFFFEF03F3F3F3F7E7E7
EFCFCF8F0E0C080"
1570 Z=10 :: T=16 :: GOSUB 1730 :: L=G ::
 O=J :: C=6 :: A=INT(RND*2)+1 :: ON A G
OTO 1580,1600
1580 A0=INT(RND*6)+1 :: IF A0=A1 THEN 158
0 ELSE A1=A0
1590 ON A1 GOSUB 2270,2280,2290,2300,2310
,2320 :: GOTO 960
1600 A3=INT(RND*6)+1 :: IF A3=A2 THEN 160
0 ELSE A2=A3
1610 D=9 :: IF A3=6 THEN GOSUB 1670 ELSE
GOSUB 1680
1620 ON A3 GOSUB 2330,2340,2350,2360,2370
,2380 :: GOTO 960
1630 Z=16 :: T=2 :: GOSUB 1730 :: L=G ::
O=J :: C=15 :: D=2 :: GOSUB 1690 :: U0=
INT(RND*3)+1 :: IF U0=U THEN 1630 ELSE U=
U0
1640 ON U GOSUB 2390,2400,2330 :: GOTO 96
0
1650 A$="0101030307070F0F" :: B$="1F1F3F3
F7F7FFFFF" :: C$="8080C0C0E0E0F0F0" ::
D$="F8F8FCFCFEFEFFFF"
1660 Q$=V$&V$&A$&B$ :: R$=A$&B$&P$&P$ ::
T$=C$&D$&V$&V$ :: U$=P$&P$&C$&D$ :: RET
URN
1670 RESTORE 700 :: GOTO 1720
1680 RESTORE 630 :: GOTO 1720
1690 RESTORE 540 :: GOTO 1720
1700 RESTORE 590 :: GOTO 1720
1710 RESTORE 610 :: GOTO 1720
1720 READ W$,CX$ :: CALL CHAR(40,CX$):: C
ALL SPRITE(#1,40,D,G,J):: RETURN
1730 CALL COLOR(0,1,T):: CALL CHAR(J,Q$,K
,R$,N,T$,P,U$)
1740 CALL SPRITE(#25,J,Z,F,I,#26,K,Z,H,I,
#27,N,Z,F,N,#28,P,Z,H,N)
1750 RETURN
1760 FOR X=15 TO 20 :: FOR Y=11 TO 17 ::
CALL HCHAR(Y,X,31):: NEXT Y :: NEXT X :
: RETURN
1770 SUB ACX(AT$,MT$)
1780 Q=INT(RND*57)+1
1790 IF Q>55 THEN 1830 ELSE 1800
1800 IF Q>28 THEN 1820 ELSE 1810
1810 ON Q GOSUB 1840,1850,1860,1870,1880,
1890,1900,1910,1920,1930,1940,1950,1960
,1970,1980,1990,2000,2010,2020,2030,2040,
2050,2060,2070,2080,2090,2100,2110 :: G
OTO 2410
1820 ON Q-28 GOSUB 2120,2130,2140,2150,21
60,2170,2180,2190,2200,2210,2220,2230,2
240,2250,2260,2270,2280,2290,2300,2310,23
20,2330,2340,2350,2360,2370,2380 :: GOT
O 2410
1830 ON Q-55 GOSUB 2390,2400 :: GOTO 2410
1840 RESTORE 160 :: RETURN
1850 RESTORE 170 :: RETURN
```

```
1860 RESTORE 180 :: RETURN
1870 RESTORE 190 :: RETURN
1880 RESTORE 200 :: RETURN
1890 RESTORE 210 :: RETURN
1900 RESTORE 220 :: RETURN
1910 RESTORE 230 :: RETURN
1920 RESTORE 240 :: RETURN
1930 RESTORE 250 :: RETURN
1940 RESTORE 260 :: RETURN
1950 RESTORE 270 :: RETURN
1960 RESTORE 280 :: RETURN
1970 RESTORE 290 :: RETURN
1980 RESTORE 300 :: RETURN
1990 RESTORE 310 :: RETURN
2000 RESTORE 320 :: RETURN
2010 RESTORE 330 :: RETURN
2020 RESTORE 340 :: RETURN
2030 RESTORE 350 :: RETURN
2040 RESTORE 360 :: RETURN
2050 RESTORE 370 :: RETURN
2060 RESTORE 380 :: RETURN
2070 RESTORE 390 :: RETURN
2080 RESTORE 400 :: RETURN
2090 RESTORE 410 :: RETURN
2100 RESTORE 420 :: RETURN
2110 RESTORE 430 :: RETURN
2120 RESTORE 440 :: RETURN
2130 RESTORE 450 :: RETURN
2140 RESTORE 460 :: RETURN
2150 RESTORE 470 :: RETURN
2160 RESTORE 480 :: RETURN
2170 RESTORE 490 :: RETURN
2180 RESTORE 500 :: RETURN
2190 RESTORE 510 :: RETURN
2200 RESTORE 520 :: RETURN
2210 RESTORE 530 :: RETURN
2220 RESTORE 560 :: RETURN
2230 RESTORE 570 :: RETURN
2240 RESTORE 580 :: RETURN
2250 RESTORE 600 :: RETURN
2260 RESTORE 620 :: RETURN
2270 RESTORE 640 :: RETURN
2280 RESTORE 650 :: RETURN
2290 RESTORE 660 :: RETURN
2300 RESTORE 670 :: RETURN
2310 RESTORE 680 :: RETURN
2320 RESTORE 690 :: RETURN
2330 RESTORE 710 :: RETURN
2340 RESTORE 720 :: RETURN
2350 RESTORE 730 :: RETURN
2360 RESTORE 740 :: RETURN
2370 RESTORE 750 :: RETURN
2380 RESTORE 760 :: RETURN
2390 RESTORE 770 :: RETURN
2400 RESTORE 780 :: RETURN
2410 READ LT$
2420 IF LT$=MT$ OR LT$=AT$ THEN 1780 ELSE
 AT$=LT$
2430 SUBEND
```

Un double face sur votre TI

Sophie Ehster

Oui vous avez bien lu : il est possible d'installer un lecteur de disquettes double face sur notre TI 99. Il y a tout de même un petit "mais". En effet, la petite modification "hardware" que je vous propose consiste à créer deux unités logiques sur la même unité physique. Expliquons nous : une unité physique c'est le lecteur au sens matériel du terme alors que son équivalent logique est celui que perçoit le système, c'est-à-dire de simples canaux d'entrées/sorties.

Ainsi, quand vous copiez à l'aide du DISK MANAGER un fichier ou une disquette complète avec un seul lecteur de disquettes, le système considère qu'il a affaire à deux unités logiques distinctes (il vous demande de changer de disque aussi souvent que nécessaire) alors qu'il n'y a bien qu'une seule unité physique.

Notre démarche est inverse : partant d'un lecteur de disquettes double face nous allons forcer le TI à condidérer chacune d'elles comme un lecteur indépendant.

Le produit obtenu ne mérite en fait pas l'appellation *double face* car il n'est pas entièrement transparent vis à vis de l'utilisateur. Sur un vrai double face, on ne se rend compte de rien; on ne peut savoir sur quelle face sont situés les programmes, ni quand le lecteur accède à l'une des faces. C'est exactement comme si on avait une seule face de capacité double.

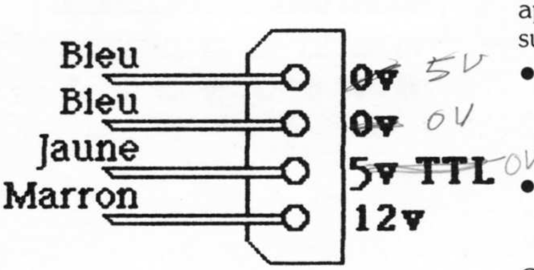
Dans le cas présent, et par la force des choses, tout se passe comme si l'on avait deux lecteurs de disquettes totalement indépendants. Notons tout de même que les topologies des deux faces sont différentes et que, par conséquent, on ne peut retourner la disquette lors de son introduction.

Cette petite modification apporte à mon sens deux énormes avantages :

le gain de place (ce qui n'est pas négligeable avec un système aussi volumineux que notre Texas) et surtout le prix.

Sachez en effet que l'on peut pratiquement raccorder n'importe quel lecteur double face au Texas (à part quelques originaux comme APPLE, pour ne pas les citer, tous les lecteurs de disquettes ont les mêmes connecteurs, tant pour l'alimentation que pour les commandes). A ce sujet, un petit rappel s'impose; la capacité d'une disquette ne dépend nullement du lecteur (qui possède néanmoins une limite supérieure de par sa mécanique) mais du contrôleur qui gère l'ensemble. Ainsi, si le lecteur X fonctionnant sur l'ordinateur Y permettait de stocker 400K par disquette, il n'en archivera toujours que 90 (hélas !) une fois raccordé au TI-99.

Schéma 1



Pour ma part, j'ai acheté un lecteur double face destiné au TRS-80 pour environ 2000 F. Moralité : pour un prix inférieur à celui pratiqué par Texas Instruments pour son simple lecteur de disquettes interne (d'ailleurs pratiquement introuvable sur le marché), on peut se payer luxueusement un double face. Je vous laisse le soin de chiffrer l'économie réalisée; sachez simplement qu'un lecteur externe se vendait aux alentours de 5000F.

Une dernière précision avant de détailler la procédure à suivre pour l'installation : si vous ne vous sentez pas d'attaque pour réaliser la petite manipulation dont je vais vous parler, ou si vous êtes réfractaire à toute forme de "bidouille", sachez qu'il est parfaitement possible d'installer le lecteur SANS AUCUNE MODIFICATION, et que le système le considérera alors comme un simple face, ignorant royalement l'autre (il va sans dire qu'il est préférable, dans ce cas, d'acheter un simple face). Ceci devrait mettre un terme à la pénurie de lecteur de disquettes.

Un dernier mot; le schéma décrit ciaprés s'applique à la configuration suivante :

- substitution du lecteur interne de chez Texas par un double face (qui jouera donc les rôles de DSK1 et de DSK2);
- transformation de l'ancien lecteur en DSK3 par adjonction d'une alimentation.

C'est un point qu'il convient d'éclaircir. Bien que Texas Instruments ait prétendu le contraire (pour des raisons que vous allez vite comprendre) les lecteurs internes et externes sont RIGOUREUSEMENT IDENTIQUES dans leur logique et dans leur mécanique. Les premiers coûtent 2500 F, les seconds le double (pour un petit transformateur et un petit coffret en plus). Tout ça pour dire qu'il suffit de fabriquer une petite alimentation (ou de la faire fabriquer) pour recycler un lecteur interne en externe.

Les petits malins vont essayer de repiquer l'alimentation du lecteur interne; erreur ! J'ai essayé, ça ne marche pas : les tensions chutent au bout d'une trentaine de secondes.

Remarque : le boîtier contient en fait deux alimentations :

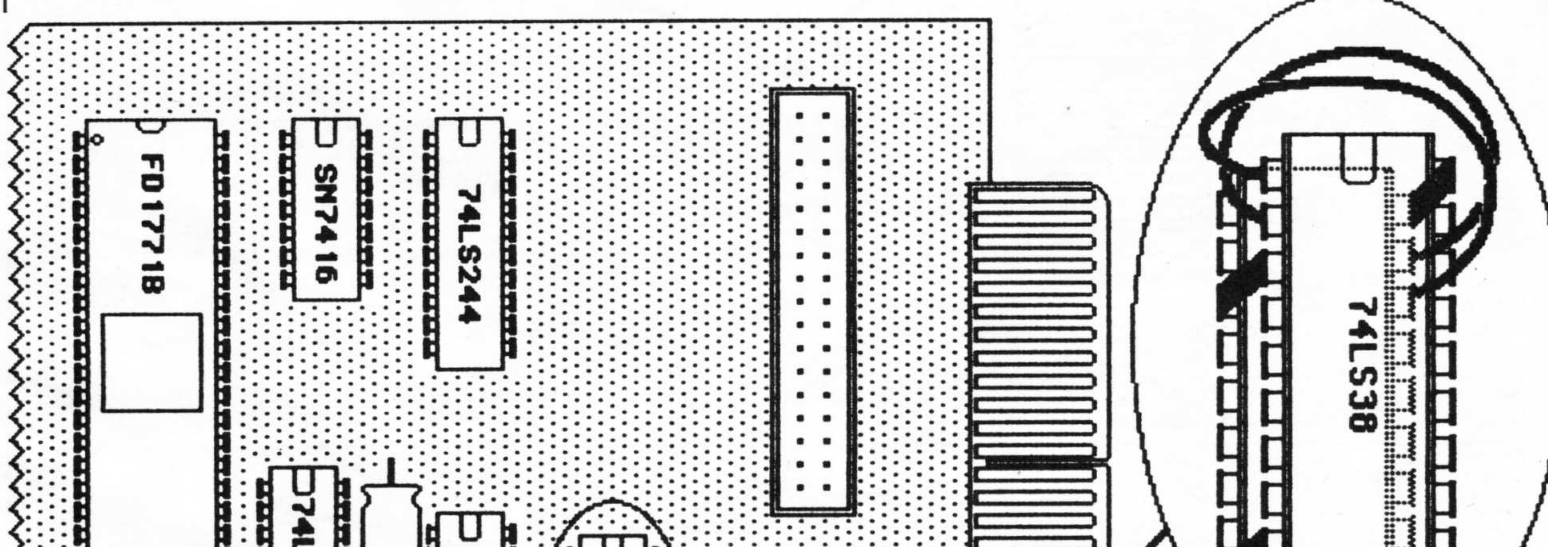
- l'une destinée au lecteur de disquettes, qui est donc dimensionnée pour UN SEUL appareil
- l'autre destinée à alimenter les différentes cartes.

Cette dernière a vraisemblablement dû être largement dimensionnée et permettrait peut-être d'alimenter un lecteur; faute de temps, je n'ai pu essayer cette opération. Si vous y arriviez et que ça marche, faites le nous savoir.

Pour ma part, je me suis fait fabriquer (pour 100 F de composants) une petite alimentation externe dont les caractéristiques sont les suivantes : 25W, 2 sorties régulées, l'une à 12v et l'autre à 5v TTL. Le câblage de l'alimentation sur la carte du lecteur est donné par le schéma 1.

Sachez enfin que pour relier un exlecteur interne devenu externe au connecteur arrière de la carte contrôleur, on peut mettre bout à bout deux des câbles de liaison fournis par Texas Instruments avec chaque boî-

Schéma 2



tier. Il est nécessaire d'en mettre deux car ces câbles inversent certaines broches, alors que dans le cas présent il faut les relier directement. La double inversion s'annulant, tout rentre dans l'ordre. Notons que pour raccorder le câble au lecteur, il faut détruire le détrompeur du connecteur (du câble) et l'insérer correctement dans le lecteur, en laissant le fil rouge du câble sur la droite, si l'on regarde le lecteur de face (ouf!).

Pour en finir avec ces liaisons, si tout ceci vous paraît bien compliqué, allez chez un revendeur et achetez le câble adéquat (tout bête et tout simple...).

Revenons donc, après toutes ces considérations annexes, au coeur du problème : comment gonfler notre simple face un peu asthmatique en un double face turbo.

Première étape, le matériel : 10 centimètres de fil électrique très fin (1/10 de mm), un fer à souder à panne très fine (il s'agit malgré tout d'un travail demandant une certaine précision) et un circuit intégré *74 LS 38*, en vente chez tous les marchands de composants pour moins de 10 F.

Cette opération à coeur ouvert va se dérouler en deux temps. Primo, adjonction du circuit intégré *74LS38* à la carte contrôleur comme décrit sur la figure 2. Secundo, soudure d'un fil électrique (shunt) entre les pattes *12* et *32* du connecteur du lecteur, aprés avoir isolé cette dernière comme indiqué en figure 3. Voyons en détail chacune des deux opérations.

Commençons donc par relier entre elles, et conformément à la figure 2, les différentes pattes du *74LS38* (2 petits bouts de fils électriques conviennent parfaitement). Rappelez vous (nous l'avons déjà mentionné dans un précédent numéro) de débrancher votre fer avant de faire la soudure. Ceci fait, pliez (ou coupez délicatement) les pattes qui ne servent à rien, puis soudez le circuit sur celui indiqué. A moins de séparer les deux circuits de plus de 5 millimètres, cette superposition n'empêche aucunement le contrôleur de réintégrer son boîtier. La première partie de l'opération est terminée.

A ce stade, deux possibilités :

- ou vous êtes vraiment très maladroit et votre contrôleur est cliniquement mort;
- ou il sait désormais gérer un double face.

Il ne reste donc plus qu'à modifier le lecteur. Pour supprimer le rôle de la patte *32* on peut soit couper le contact en cuivre qui part de la patte, soit recouvrir cette dernière d'un bout de ruban adhésif pour l'isoler du connecteur. Soudez enfin une dérivation allant de la patte *12* à la patte *32*, et modifier le support des shunts (il s'agit de l'élément situé en haut à droite de la carte du lecteur, juste derrière le fameux "resistor pack") conformément a la figure 4.

Schéma 3

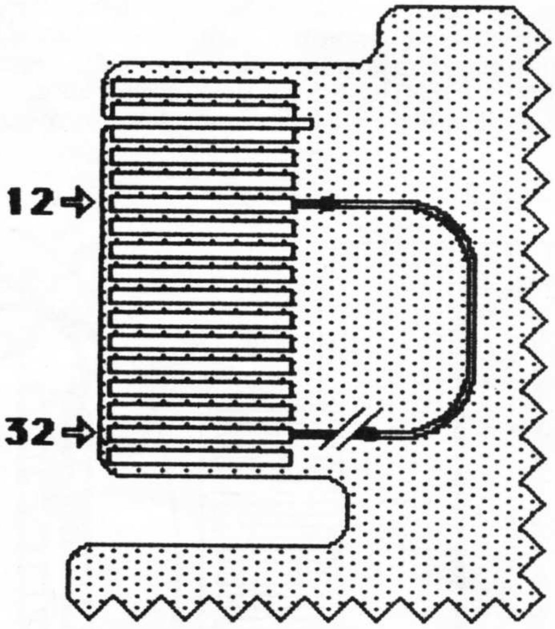
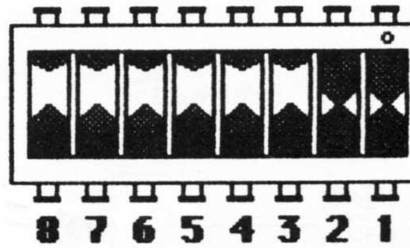
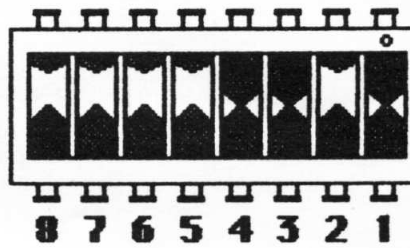


Schéma 4

Interne faisant office de 1 et 2



Externe DSK3



Votre système est désormais prêt. Tapez *OLD DSK2.ZOULOU* et Oh miracle de l'électronique, la diode de DSK1 s'allume. Il ne vous reste plus qu'à initialiser votre face cachée pour pouvoir vous en servir.

Si un utilisateur arrive à faire marcher un triple face qu'il nous écrive, nous serions heureux de le publier.

■

Courrier des lecteurs

Alexandre Duback

Certains de vos programmes, pour ne pas dire tous, comportent des erreurs inexplicables. Par exemple, dans le numéro 4 page 9, programme "Formule 1", le programme a accepté de tourner uniquement après avoir modifié la ligne 220, soit FOR I = 1 TO 60 ?.

Albert Belou - 23 bis, rue Vauban - 31000 Toulouse.

Les programmes publiés dans 99 Magazine sont tous soigneusement vérifiés avant publication. De plus, les listings qui apparaissent dans la revue sont imprimés en dernier lieu, ce qui fait que les risques d'erreurs sont très faibles.

En ce qui concerne le programme mis en cause dans votre lettre, nous pouvons vous affirmer qu'il fonctionne normalement. Ce programme comporte beaucoup de "DATA" et nous savons par expérience que cette instruction est la principale source d'erreurs lors de la saisie (il est très facile de remplacer un zéro par un "O" ou d'oublier une virgule).

J'ai tapé sur mon ordinateur le jeu "Trouvez le bon mot" mais, malheureusement, je ne peux pas l'utiliser. En effet, à chaque mot imprimé, j'obtiens le message "Aucune lettre" et à la solution, je découvre qu'en fait le message aurait dû être "Une lettre" ou "Deux lettres", etc... J'ai essayé de trouver l'erreur et je pense que c'est à la ligne 2020 que le programme prend la direction "ligne 2030". Cela voudrait dire que E(LX) est toujours égal à 0. Je n'est pas été capable de pousser mes recherches plus loin et j'aimerais que vous apportiez une solution à ce problème, ce qui me permettrait d'utiliser ce jeu qui me paraît intéressant.

J'aimerais également savoir si vous vérifiez le matériel proposé dans les petites annonces. J'ai lu dans le numéro 5 qu'une personne vend un contrôleur et un lecteur de disquette, mais ceci dans un lot complet de 9000 F. Pouvez-vous me donner un prix aproximatif de ce matériel d'occasion.

Jean-Luc Delaunay - 54 Avenue Foch - 76290 Montivilliers

Votre première question appelle la même réponse que pour la lettre précédente. Nous vous invitons donc à vérifier votre programme car il fonctionne parfaitement bien, tous nos lecteurs ayant fait l'acquisition de la cassette d'accompagnement peuvent d'ailleurs en témoigner. A propos des cassettes d'accompagnement, et sans vouloir pousser à la consommation, il est bon de rappeler que pour un prix dérisoire (elles sont toujours vendues 55 F malgré les coûts de production qui ont plus que doublés en un an) nous vous évitons des heures de saisie et de recherche d'erreurs.

Au sujet des petites annonces, il nous est matériellement impossible de contrôler ce que propose nos lecteurs, et cela pour une raison évidente : nous sommes à Versailles, si quelqu'un vend un module "Basic Etendu" à Marseille... Nous ne connaissons d'ailleurs pas de journaux qui pratiquent ce genre de sport, même lorsque les annonces sont "lucratives". Pour le prix du matériel d'occasion, il n'y a qu'une seule règle : l'offre et la demande, et il est inutile d'essayer d'établir un barème car les prix varient dans des proportions considérables. Ceci étant dit, pour le matériel proposé dans l'annonce citée dans votre lettre, 9000 F nous semble être une somme tout à fait honnête, compte tenu du prix du matériel neuf.

Après quelque peu d'hésitation, je vous adresse un bulletin d'abonnement. L'abandon par Texas du TI-99 me dissuadait de m'abonner car je craignais votre disparition. Je préférais donc acheter chaque numéro au fur et à mesure. Je vous fais donc confiance pour la suite car j'ai noté (en particulier dans les petites annonces) que beaucoup cherchent encore à se procurer du matériel TI. Cela ne m'étonne pas car utilisant un TI-99 depuis bientot deux ans, j'ai pu apprécier sa fiabilité, même en utilisation intensive.

J'emploie le TI-99 surtout pour établir mes propres programmes car je n'aime guère recopier des programmes pré-fabriqués qui concordent rarement avec mes besoins. J'ai donc créé de nombreux programmes personnels (gestion de mes affaires, comptabilité, traitement de texte, fichiers divers, etc...).

A l'instar d'autres lecteurs, je vous félicite de l'édition de tuyaux sur l'Assembleur car, n'ayant pas encore suffisamment maîtrisé son utilisation, vos renseignements m'aident beaucoup (le manuel de l'Editeur/Assembleur ne comporte pas assez d'exemples). En particulier, j'ai intégré votre sous-programme de routines graphiques (numéro 4 de "99 Magazine") à un programme d'étude de fonctions mathématiques que j'avais établi depuis assez longtemps et qui permettait aussi le tracé de courbes, mais une seule à la fois. Votre sous-programme me permet aussi l'étude de fonctions paramétriques par superposition de plusieurs graphiques. J'ai aussi tapé à titre d'entraînement (11 pages... Ouf !) le programme de Master Mind (votre numéro 3) qui fonctionne de façon parfaite à la grande joie de mes enfants.

En vous encourageant fortement à continuer très longtemps la parution de votre intéressante revue, je vous prie de croire , Messieurs, à mes meilleurs sentiments.

Jean Simoni - 77, rue Lafayette - 94210 La Varenne.

Cette aimable lettre constitue une réponse au deux précédentes et prouve, si besoin est, que les listings publiés dans 99 Magazine sont dépourvus d'erreurs, d'autant plus que les programmes "Master Mind" et "Routines graphiques" sont incomparablement plus complexes que les programmes "Trouvez le bon mot" et "formule 1". Merci, Monsieur Simoni...

Je suis heureux d'être abonné à votre revue et je souhaite que vous transmettiez mes félicitations et mes remerciements à toute l'équipe de rédaction.

Dans les numéros 4 et 5, vous développez un article sur le graphisme en haute résolution. Comme Gérard Santraille et Jean-Luc Bazanegue nous y invitent, j'ai "accroché ma ceinture...". Malheureusement, je ne possède pas le module Mini-mémoire ou l'Editeur/Assembleur. Dans ces conditions, vous serait-il possible de prolonger cet article pour la configuration suivante : Basic Etendu, carte 32 Ko et lecteur de disquettes.

Toujours dans le cadre des graphismes, vous serait-il possible de m'indi-

quer la façon d'obtenir des copies d'écran sur Seikosha GP100.

Alain Crozet - 15, rue B. Palissy - 16340 L'Isle d'Espagnac.

Les routines graphiques ne peuvent malheureusement pas fonctionner avec le Basic Etendu, et ceci pour une raison très simple : ce module utilise la partie haute de l'extension de mémoire pour le stockage du programme Basic, qui viendrait écraser les routines en assembleur et le tampon "TIMAGE". Nous n'avons pas encore trouvé la solution pour "faire croire" au Basic Etendu que l'extension de mémoire n'est pas présente. La seule solution que nous puissions vous proposer est d'acquérir un module Mini-mémoire; ceci est à nouveau possible car "La règle à calcul" a pris l'initiative de reprendre la fabrication des modules "Mini-mémoire" et "Basic Etendu" ainsi que d'une interface parallèle qui ne nécessite pas le boîtier d'extension.

Le programme de copy d'écran sur Seikosha, publié dans ce numéro devrait vous satisfaire.

Je suis possesseur d'un TI-99/4A acquis en novembre 1983. Lors de cette acquisition, le vendeur m'avait dit que je bénéficiais d'un abonnement gracieux à "99 Magazine". J'ai effectivement reçu - et apprécié - les trois premiers numéros de ce magazine. Je vous serais reconnaissant de m'adresser les numéros suivants et vous en remercie d'avance.

Albert Plantier - 8, rue du Chatanay - 38230 Tignieu.

Nous vous remercions pour l'intérêt que vous portez à notre revue mais permettez nous de mettre les choses au point : Texas instruments France nous avait acheté des exemplaires des numéros 1, 2 et 3 afin de les remettre à leurs clients. Lors de l'arrêt de la commercialisation du TI, le nombre d'exemplaires à envoyer gratuitement était supérieur au nombre d'appareils restant à vendre; ceci explique que vous ayez reçu trois numéros, alors que le bon accompagnant le TI lors de l'achat spécifiait "Pour obtenir GRATUITEMENT un exemplaire de cette revue...". Il est possible que certains revendeurs aient utilisé cet "abonnement fantôme" comme argument de vente mais si tous les utilisateurs de TI étaient abonnés gratuitement à "99 Magazine", il est fort probable que nous ne surviverions pas longtemps.

Existe-t-il une méthode pour accéder à une autre partie de la mémoire (2000 à 3FFF ou A000 à FFFF) sans extension de mémoire.

Emile Fraysse - 25, Avenue Emile Dewditine - 31200 Toulouse

Ces adresses étant situées physiquement dans l'extension 32 Ko, vouloir y accéder reviendrait à demander à un facteur de poster une lettre au numéro 50 d'une rue ne comportant de 25 habitations.

Est-il possible de concaténer des "sous-chaînes" de caractères de longueur finie (constantes) se terminant par des "blancs" ? Je n'ai pas réussi et ai réglé le problème en terminant chaque sous-chaîne par un point.

Jean-Pierre Boisnard - 163, Avenue de Grandre Bretagne - 31300 Toulouse

Nous pensons que votre question porte sur les chaînes de caractères sous forme de DATA. Dans ce cas, pour que les espaces précédant et suivant la chaînes soit reconnus, il faut que la chaîne soit fermée par des guillemets. ■

Petites annonces (gratuites)

Vends module "Echecs en français, ou échange contre module "Terminal Emulator II".

Achète boîtier d'extension avec, au minimum, la carte RS232. Bernard Gibon - 5, rue des frères Morane - 75015 Paris - 828.73.10 le soir.

Recherche lecteur de disquettes et contrôleur. Alain Crozet - 15, rue B. Palissy - 16340 L'Isle d'Espagnac.

Vends TI-99/4A, Boîtier d'extension, carte 32 Ko, lecteur de disquettes, Basic Etendu, Logo et nombreux autres modules, documentation, livres et programmes. C. Duchenoy - 1, Avenue de la république - 94380 BONNEUIL - 339.15.08 le soir.

Cherche boîtier d'extension, carte 32 Ko, carte RS232, contrôleur et lecteur de disquettes, imprimante, Editeur/Assembleur, TI-Calc. (99) 68.99.45.

Vends modules Chisholm trail, Wumpus, Meteor multiplication, Alien addition, Démolition division. 120 F pièce ou 550 F le tout. Crozet - 15, rue B. Palissy - 16340 L'Isle d'Espagnac.

Recherche 2 modules "Speech editor" et 2 carte contrôleur de disquettes. Centre médical de rééducation logopédique - Square Roosevelt, 2 - 7000 Mons - (065) 33.12.26.

Achète interface RS232 et donne en supplément programmes au choix. Eric Lafeil - 10, Chemin de la gouttière - 91310 Montlhéry - (16-6) 901.50.17.

Vends 600 F Editeur/Assembleur avec manuel "Initiation au langage assembleur du Texas Instruments TI-99/4A". Laurent De Bellis - 16, rue Hector Malot - 69007 Lyon - (7) 869.21.74 le week end.

Recherche module Basic Etendu. Morlon - 80, rue de la République - 54140 Jarville - (8) 354.20.59 après 20 h.

Vends TI-99/4A. Modules Parsec, Invaders, Hustle, Star Trek, Echecs, Music Maker. Manettes de jeux, cordon magnéto, 2 livres programmes et 3 cassettes 99 Magazine. Le tout en très bon état pour 2800 F. Pierre Petitjean - 7, rue d'Alembert - 18000 Bourges - (48) 65.62.54.

Recherche programmes sur K7 ou listing pour échanges prolongés. Michel Hanne - 16, rue du Cura - 1400 Nivelles - Belgique.

Vends TI-99/4A acheté 9/83, manuel, cordon magnéto, manettes, nombreux programmes et livres. Le tout 1500 F. G. Burdallet - Quartier du Roure - 13370 Mallemort.

Vends TI-99, magnétophone Philips, cordon magnétophone, Parsec, Retour du pirate, jeux "rétro II", et différents programmes et livres. 2500 F. Stéphane Battistella - 75 bis rue Michel-Ange - 75016 Paris - (1) 651.34.91 après 18 h.

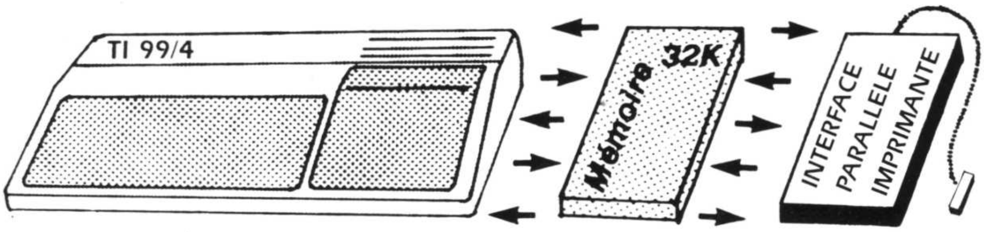
■

la Règle à Calcul

A vos consoles T.I. 99/4, basic étendu disponible

LA RÈGLE A CALCUL

sort une extension mémoire 32 K extérieure pour la console TI 99/4.
Avec 48 K ram, vous allez réaliser des programmes plus performants.



BRANCHEMENT IMMÉDIAT EXTENSION 32 K 1 340 F
INTERFACE IMPRIMANTE 1 090 F

PROMOTION
LIVRAISON OCTOBRE

LOT N° 1
Module basic étendu
+
Interface parallèle centronics
+
Imprimante Seikosha GP 500 colonnes
L'ensemble : 4 290 F TTC

LOT N° 2
Module basic étendu
+
Extension mémoire ext. 32 K
+
Interface parallèle centronics pour TI 99
+
Imprimante EPSON RX 80
L'ensemble : 6 560 F TTC

GESTION LOT N° 3
Module basic étendu
+
Basic par soi-même
+
Gestion privée
Aide à la programmation
L'ensemble : 1 205 F TTC

LOISIRS LOT N° 4
Module basic étendu
+
Retour du Pirate
+
Startreck
+
Introduction au jeux 1
+
Introduction au TI 1
+
Basic par soi-même
L'ensemble : 1 425 F TTC

LOISIRS LOT N° 5
Module basic étendu
+
Munch man
+
Invaders
+
Jeux vidéo 2
+
Jeux rétros 2
+
Introduction aux jeux 1 et 2
L'ensemble : 1 540 F TTC

GESTION LOT N° 6
Module basic étendu
+
Conseils financiers
+
Gestion fichiers
+
TI Calc
+
Gestion privée
L'ensemble : 1 755 F TTC

LOISIRS LOT N° 7
Module basic étendu
+
SOFTWARE HEBDOGICIELS
TI n° 2 basic étendu 12 jeux
+
Rubis sacré (jeu d'aventures)
+
Lunar lander 2 (pilote spatial 25 étapes)
+
Introduction au TI 99 1
+
Technique programmation de jeux 1
L'ensemble : 1 185 F TTC

ACCESSOIRES TI 99/4	PRIX TTC	
Modulateur SECAM France	500,00	☐
Câble liaison magénto-cassettes	100,00	☐
Paire manettes jeux	210,00	☐
Magénéto-cassettes compatible T.I. LANSAY avec compteur + câble magnéto compris	370,00	☐
Imprimante GP5OA papier thermique	1.305,00	☐
EPSON RX 80	3.440,00	☐
Imprimante Seikosha GP 500	2.500,00	☐
Interface parallèle extérieure	1.090,00	☐
PROGRAMMATION		
Extended basic module USA manuel français.	900,00	☐
Mémoire extension 32 K extérieure	1.340,00	☐
Basic par soi-même	70,00	☐
Aide à la programmation	75,00	☐
ORGANISATION		
Conseil financier	75,00	☐
Gestion fichier	260,00	☐
Gestion rapports	375,00	☐
Statistics	206,00	☐
Ti calc	360,00	☐
Gestion privée	360,00	☐
MODULES LOISIRS		
Connect four	147,00	☐
Munch Man	252,00	☐
Othello	206,00	☐
The attack	147,00	☐
Jeux Rétro II	147,00	☐
Ti-invaders	206,00	☐
Jeux vidéo II	147,00	☐
Jeux Rétro I	147,00	☐
NOUVEAUX LOGICIELS TEXAS		
Retour pirate	250,00	☐
Demon attack	250,00	☐
Mash	250,00	☐
Burger time	250,00	☐
Hopper	250,00	☐
Star treck	250,00	☐
Jaw breaker	250,00	☐
Treasure Island	250,00	☐
Microsurgeon	250,00	☐
Moonsweeper	250,00	☐
PROGRAMMES COLLINS/TEXAS		
Introd. au TI 99/4 (1)	75,00	☐
Introd. au Ti 99/4 (2)	75,00	☐
Les techniques des programmes de jeux (1)	75,00	☐
Les techniques des programmes de jeux (2)	75,00	☐
Lunar lander 2 (pilote spatial 25 étapes)	95,00	☐
HEBDOGICIELS SOFTAWARE		
TI N° 1 Basic simple K7 : 12 jeux	150,00	☐
TI N° 2 Basic étendu : 12 jeux	150,00	☐
TI Rubis sacré Jeu d'aventures basic étendu	90,00	☐

ATARI SOFT		
Pole position	350,00	☐
Moon patrol	350,00	☐
Jungle hunt	350,00	☐
Defender	350,00	☐

BON DE COMMANDE

TARIFS OCTOBRE 1984

Nom ..
Prénom ..
Adresse ..
.. *Tél. :*
Code Postal ..
Ville ..

Ces prix sont indicatifs et peuvent être modifiés sans préavis.

Produits disponibles dans la limite de nos stocks en magasin.

Participation aux frais de port et d'expédition en recommandé pour les logiciels : + 30 F.

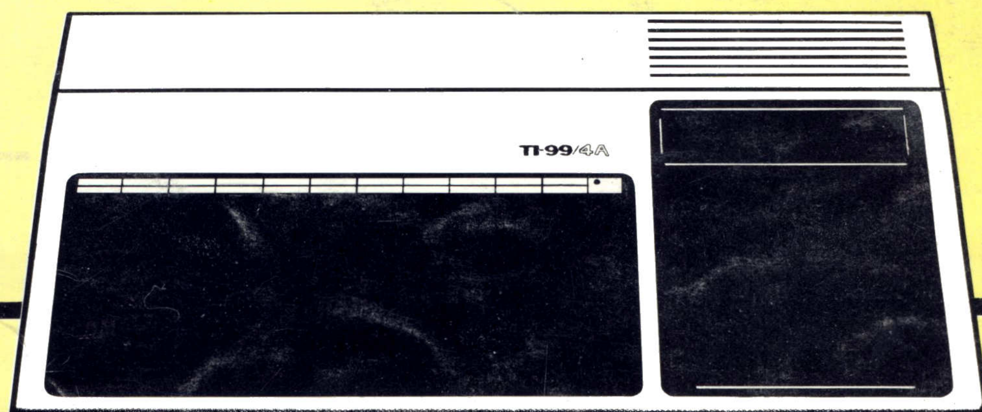
LA RÈGLE A CALCUL :
65/67, bd Saint-Germain, 75005 PARIS
Tél. : 325.68.88 - Télex : ETRAV 220064F/1303 RAC.
Livraison des produits disponibles sous 8 jours.

Texas Instruments vient de céder sa licence de fabrication de la console TI 99 à une firme japonaise qui compte sortir ce produit pour la fin de l'année.
Aux U.S.A., une étude de contrôleurs et d'unités de disques pour TI 99 double face, double densité, est prévue pour fin septembre utilisation externe. Des essais sont en cours en R.F.A. pour utiliser ce contrôleur dans le système RAC intégré.

CASSETTE
D'ACCOMPAGNEMENT
DISPONIBLE

99 MAGAZINE

COPIE D'ECRAN EN ASSEMBLEUR
PASCAL : LES BIBLIOTHEQUES
BASIC TI — BASIC ETENDU
LOGO ET LA RECURSIVITE...



Numéro 7 - Trimestriel - Décembre 1984 - 40 F

ISSN 0759-9005

99 MAGAZINE

Sommaire

	Page
Editorial par Hervé Thiriez	5
La Défausse par Edouard Krok	6
Le tableur TI-Calc par Sophie Ehster	10
Sauvegarde des images graphiques et copie d'écran par Jean-Luc Bazanegue	11
Copie d'écran sur Seikosha GP250 par Michel Schurmans	22
La compilation séparée - Les bibliothèques par Gérard Santraille	23
Trouvez votre chemin dans les jeux d'aventure TI par Denise Amrouche	25
Srand par Georges Goument	28
Galaxia par Pierre Marcellin	34
Logo et la récursivité par Michel Schurmans	50
Dessinez avec le Basic Etendu par Georges Herbepin	51
Accès au mode Bit-Map en Basic TI par Gérard Baroni et Jean Marin	54
Music Box Dancer par Jacques Mercier	55
Erratum	59
Des nouvelles du club Ticazur	59
Routines graphiques pour le Basic Etendu par Marianne Sutz	60
Courrier des lecteurs par Alexandre Duback	63
Petites annonces	65

Les annonceurs

La règle à calcul : page 2 - LIST : page 67 - Atari : page 68.

Editions MEV - 64-70, rue des Chantiers - 78000 Versailles

(Dépôt légal : 4e trimestre 1984)

Editorial

*Cette fois encore, **Edouard Krok** nous propose un de ses excellents programmes en Basic TI. Il s'agit d'un jeu de cartes dans lequel le TI peut tenir le rôle de trois joueurs. **Georges Goument** s'est chargé de l'écriture d'un autre jeu qu'il a baptisé "Srand".*

*Les tableurs étant très à la mode, **Sophie Ehster** nous donne son avis sur TI-Calc, alors que **Gérard Santraille**, toujours très attaché au Pascal, dévoile les secrets de la compilation séparée et des bibliothèques.*

*Les amateurs d'assembleur auront de quoi occuper les trois mois nous séparant du prochain numéro puisque **Jean-Luc Bazanegue** propose une routine de codage et sauvegarde des images graphiques en haute résolution, fort attendue par les utilisateurs des routines graphiques. Dans un tout autre registre, mais toujours en assembleur, vous trouverez aussi "Galaxia", écrit par **Pierre Marcellin**, qui est un programme inspiré de TI-Invaders mais, à notre avis, largement supérieur. En outre, **Denise Amrouche** s'est penché sur le problème des jeux d'aventure et a découvert le moyen de connaître leurs contenus.*

***Michel Schurmans** reprend la rubrique Logo, ce qui fera certainement plaisir aux fanatiques de ce langage puisque nous les avions quelque peu délaissés dans le précédent numéro. Le même auteur offre aussi un programme de copie d'écran aux utilisateurs d'imprimantes Seikosha GP250.*

*Après nous avoir indiqué comment accéder aux registres vidéo à partir du Basic TI, **Gérard Baroni et Jean Marin** donnent la méthode pour utiliser le mode Bit-Map de la même manière. Ceux d'entre-vous qui désiraient employer les routines graphiques depuis le Basic Etendu pourront désormais le faire puisque **Marianne Sutz** a effectué les modifications nécessaires.*

*Enfin, "Music Box Dancer", de **Jacques Mercier** et "Dessinez avec le Basic Etendu", de **Georges Herbepin**, bouclent ce numéro qui, vous pouvez le constater, contient de quoi occuper vos loisirs informatiques pendant un certain temps.*

Nous vous souhaitons une excellente année 1985, pleine de programmes et, si possible, sans bugs !

Hervé Thiriez

Rédacteur en chef-Directeur de la publication : Hervé Thiriez. **Comité de rédaction** : Jean-Luc Bazanegue, Gérard Santraille. **Dessins** : Laurent Bidot. **Ont collaboré à ce numéro** : Denise Amrouche, Gérard Baroni, Jean-Luc Bazanegue, Alexandre Duback, Sophie Ehster, Georges Goument, Georges Herbepin, Edouard Krok, Pierre Marcellin, Jean Marin, Jacques Mercier, Gérard Santraille, Michel Schurmans, Marianne Sutz.

Editions MEV - 64-70, rue des chantiers - 78000 Versailles.

Régie publicitaire : Force 7 - Anne Jourdan - 5, Place du Colonel Fabien - 75010 Paris - Tél. : 240.22.01.

Impression : S.N.E. Repro-Versailles - 21, rue Danièle Casanova - 78210 St Cyr l'école - Tél. : 460.38.80.

La Défausse

Edouard Krok

NDLR : nos lecteurs munis de lecteurs de disquettes devront utiliser la commande "CALL FILES (1)" suivie de "NEW", avant de charger ce programme en mémoire.

La défausse est un jeu de cartes simple et original, écrit en **Basic TI**, qui se pratique avec un jeu de 52 cartes (plusieurs jeux de 52 cartes si on joue à 4). Chaque participant reçoit sept cartes au début de la première manche, quatre cartes à la seconde, puis une seule carte à la troisième; ensuite, on recommence à sept, et ainsi de suite.

Chacun, à son tour, pose une carte qui doit être de la couleur demandée (cœur, pique, carreau ou trèfle) ou un valet (le valet remplace n'importe quelle carte) ou une carte quelconque qui soit de la même valeur que la carte de jeu (on peut, par exemple, poser un sept de pique si la carte de jeu est un sept de cœur). Dans ces deux derniers cas, on a la possibilité de demander la couleur de son choix.

Si un joueur ne peut poser aucune carte, il doit en piocher une dans le talon. S'il lui est impossible de jouer la carte piochée, il passe son tour. Le premier joueur qui n'a plus de carte gagne la première manche et s'approprie les points des cartes qui restent à tous ses adversaires.

Le joueur dont le score est égal ou supérieur à 60 points gagne la partie. Habituellement, on joue plusieurs parties; le vainqueur sera bien sûr celui qui aura remporté le plus grand nombre de parties et, en cas d'égalité, celui qui aura cumulé le plus grand nombre de points.

Vous avez la possibilité d'utiliser à votre guise les deux poignées de jeu ou le clavier : vous déplacez le curseur sous la carte que vous voulez jouer à l'aide des touches fléchées (S et D), puis vous appuyez sur "ENTER".
Pour piocher une carte, vous devrez amener le curseur sous vos initiales, puis appuyer sur "ENTER" et procéder de la même façon pour passer le contrôle au joueur suivant, dans le cas où vous ne pouvez pas jouer la carte piochée.

Le programme accepte deux, trois ou quatre joueurs et l'ordinateur peut en remplacer un ou plusieurs. On peut aussi lui demander de jouer contre lui-même. Dans tous les cas, sa stratégie élaborée en fait un adversaire qu'il ne sera pas toujours facile de vaincre.

Si vous voulez rendre le jeu encore plus passionnant, choisissez l'option "jeu fermé"; vous ne verrez plus le jeu de l'ordinateur (il vous faudra le déduire d'après ses demandes) et vous ne pourrez voir le jeu de vos adversaires humains qu'au moment où ceux-ci doivent jouer. Ce sera plus difficile, mais aussi plus intéressant.

Valeurs des cartes

- 1 à 10 = leur valeur respective;
- Dame, roi = cinq points;
- Valet = cinquante points.

□

```
10 REM ******************
20 REM *   LA DEFAUSSE   *
30 REM ******************
40 REM *    BASIC  TI    *
50 REM ******************
60 REM *                 *
70 REM *  Edouard  KROK  *
80 REM *                 *
90 REM ******************
100 RANDOMIZE
110 DIM M(3,1,5),M$(3),J$(3),SC(3)
120 DEF V=J*4+1-(J>1)*8
130 DEF H(X$)=ABS(5+((J=1)+(J=3))*33+3*PO
S(M$(J),X$,1))
140 DEF CL=INT((CRTE-1)/13)
150 DEF VL=CRTE-CL*13
160 DEF PT=-VL*(VL<11)-50*(VL=11)-5*(VL>1
1)
170 CALL CHAR(58,"002669A9292926")
180 CALL CHAR(59,"82828244428281")
190 CALL CHAR(60,"3824222222222438")
200 CALL CHAR(61,"3C2222223C282422")
210 CALL CHAR(62,"")
220 CALL CHAR(64,"00181800001818")
230 CALL CHAR(110,"44EEFEFEFE7C381")
240 CALL CHAR(118,"3810D6FEFE7C381")
250 CALL CHAR(126,"183C7EFFFF7E3C18")
260 CALL CHAR(134,"3838D6FED610387C")
270 CALL CHAR(140,"993366CC993366CC")
280 CALL CHAR(144,"003F7EEC7E3F")
290 CALL CHAR(145,"00FC7E377EFC")
300 CALL CHAR(146,"0010387CFE38386C")
310 CALL CHAR(147,"000103070F0F0E")
320 CALL CHAR(148,"0080C0E0F0F07")
330 FOR I=1 TO 5
340 CALL COLOR(9+I,VAL(SEG$("0914100705",
I*2-1,2)),16)
350 NEXT I
360 CALL COLOR(14,5,16)
370 CALL COLOR(15,5,1)
380 GOSUB 4110
390 CALL CLEAR
400 CALL SCREEN(16)
410 CALL HCHAR(11,3,145,29)
420 CALL VCHAR(4,31,146,7)
430 CALL HCHAR(3,3,144,29)
440 CALL VCHAR(4,3,146,7)
450 FOR I=1 TO 21
460 CALL HCHAR(7,6+I,ASC(SEG$("L A   D E
F A U S S E",I,1)))
470 NEXT I
480 W=110
490 FOR C=4 TO 26 STEP 4
500 FOR L=0 TO 2
510 W=W+8+(W=134)*32
520 CALL HCHAR(14+4*L,C+3*L,W)
530 NEXT L
540 NEXT C
550 GOSUB 4880
```

```
550 GOSUB 4880
560 X$="APPUYEZ SUR ""ENTER""."
570 PRINT TAB(5);X$
580 CALL KEY(3,K,S)
590 IF K<>13 THEN 580
600 CALL SCREEN(2)
610 PRINT "VOUS DEVEZ JOUER UNE CARTE  DE
 LA COULEUR DEMANDEE OU UNVALET OU UNE
CARTE DE MEME"
620 PRINT "VALEUR QUE LA CARTE DE JEU."::
"DANS CES 2 DERNIERS CAS":"VOUS DEMANDE
Z LA COULEUR":"DE VOTRE CHOIX."
630 PRINT :"VOUS POUVEZ UTILISER LES":"MA
NETTES OU LE CLAVIER @"::"DEPLACEMENT P
AR TOUCHES S ET"
640 PRINT "D, PUIS ";X$::"SI VOUS NE POUV
EZ JOUER VOUSDEVEZ PIOCHER UNE CARTE DA
NS"
650 PRINT "LE TALON EN AMENANT LE":"POINT
EUR SOUS VOS INITIALES."::::TAB(5);X$
660 CALL SCREEN(11)
670 CALL KEY(3,K,S)
680 IF K<>13 THEN 670
690 CALL SCREEN(2)
700 PRINT "LE PREMIER JOUEUR QUI N'A"::"P
LUS DE CARTE GAGNE LES"::"POINTS DES CA
RTES QUI"::
710 PRINT "RESTENT A SES ADVERSAIRES !"::
"VALEUR DES CARTES @"::"   1 a 10 @ LEU
R VALEUR":TAB(19);"RESPECTIVE"::
720 PRINT "   DAME, ROI @ 5 POINTS"::"
VALET @ 50 POINTS."::"UNE PARTIE SE JOU
E"::"EN 60 POINTS.":::TAB(4);X$
730 CALL SCREEN(13)
740 CALL KEY(3,K,S)
750 IF K<>13 THEN 740
760 REM   rejouer avec des
770 REM  joueurs differents
780 CALL SOUND(400,600,6)
790 FOR I=0 TO 3
800 J$(I)=""
810 NEXT I
820 CALL CLEAR
830 PRINT "JEU OUVERT OU FERME ? (O/F)"
840 CALL SOUND(200,800,4)
850 CALL HCHAR(23,30,63)
860 CALL KEY(3,K,S)
870 IF S=0 THEN 860
880 CALL SOUND(100,-5,6)
890 CALL HCHAR(23,30,K)
900 W=0
910 IF K=79 THEN 940
920 W=1
930 IF K<>70 THEN 850
940 PRINT :::::"COMBIEN DE JOUEURS HUMAIN
S"::"ETES VOUS ?"::TAB(9);"( 0 a 4 )  ?
"
950 CALL SOUND(200,800,4)
960 CALL KEY(3,X,S)
970 IF (X<48)+(X>52)THEN 960
980 CALL SOUND(100,-5,6)
990 CALL HCHAR(23,22,X)
1000 IF X=52 THEN 1150
1010 PRINT ::::"COMBIEN DE JOUEURS DESIRE
Z"::"VOUS QUE JE SIMULE ?"::TAB(9);"( 0
 a   )  ?"
1020 K=100-X
1030 CALL HCHAR(23,17,K)
1040 CALL SOUND(200,800,4)
1050 CALL KEY(3,Y,S)
1060 IF (Y<48)+(Y>K)THEN 1050
1070 CALL SOUND(100,-5,6)
1080 CALL HCHAR(23,22,Y)
1090 JC=X+Y<100
1100 IF X+Y=96 THEN 4080
1110 IF X+Y>97 THEN 1150
1120 CALL CLEAR
1130 PRINT "    IL N'EST PAS POSSIBLE"::T
AB(8);"DE JOUER SEUL !"
1140 GOTO 940
1150 GOSUB 4880
1160 CALL CLEAR
1170 IF X=48 THEN 1190
1180 PRINT " INSCRIVEZ VOS INITIALES @"::
:::::::::
1190 CALL SOUND(400,600,6)
1200 X=X-48
1210 Y=Y-48
1220 FOR I=1 TO X
1230 FOR J=1 TO 2
1240 CALL KEY(3,K,S)
1250 IF S<1 THEN 1240
1260 IF (K<65)+(K>90)THEN 1240
1270 CALL SOUND(100,-5,7)
1280 CALL HCHAR(16+2*I,15+J,K)
1290 S=VAL(SEG$("3120",I,1))
1300 J$(S)=J$(S)&CHR$(K)
1310 NEXT J
1320 J$(S)=CHR$(1-2*W*(X>1))&J$(S)&"@"&"
 000  "
1330 NEXT I
1340 FOR I=1 TO Y
1350 J$(VAL(SEG$("0213",I,1)))=CHR$(1+W)&
"TI"&CHR$(96+I)&"   000  "
1360 NEXT I
1370 GOSUB 4880
1380 REM   rejouer autre partie
1390 REM      memes joueurs
1400 CALL CLEAR
1410 CALL COLOR(3,2,16)
1420 CALL COLOR(4,2,16)
1430 FOR J=0 TO 3
1440 IF J$(J)="" THEN 1480
1450 FOR I=1 TO 3
1460 CALL HCHAR(V,H("X")+I+2+((J=0)+(J=2)
)*7,ASC(SEG$(J$(J),I+1,1)))
1470 NEXT I
1480 NEXT J
1490 FOR I=1 TO 27
1500 CALL HCHAR(10,3+I,ASC(SEG$("DEMANDE
 JEU  TALON  MANCHE",I,1)))
1510 NEXT I
1520 CALL VCHAR(12,20,140,3)
1530 CALL VCHAR(12,21,140,3)
1540 MCH=0
1550 REM   debut de manche
1560 FOR L=12 TO 14
1570 CALL HCHAR(L,1,32,15)
1580 NEXT L
1590 FOR L=0 TO 1
1600 FOR C=2 TO 1 STEP -1
1610 FOR W=1 TO 3
1620 CALL HCHAR(W+L*16-4*(C=1),4*C,32,23)
1630 NEXT W
1640 NEXT C
1650 NEXT L
1660 MCH=MCH+1
1670 CALL HCHAR(13,27,48+MCH)
1680 PRI$=""
1690 REM   distribution des cartes
1700 FOR I=1 TO 10-MCH*3+9*INT(MCH/3.1)
1710 FOR J=0 TO 3
1720 IF J$(J)="" THEN 1770
1730 X=-5+1*(ASC(J$(J))=1)
```

```
1740 GOSUB 4520
1750 X=1
1760 GOSUB 4780
1770 NEXT J
1780 NEXT I
1790 X=0
1800 GOSUB 4520
1810 DEM=CL
1820 DVL=VL
1830 CALL HCHAR(13,7,110+8*CL)
1840 REM  debut de chaque tour
1850 FOR J=0 TO 3
1860 P=0
1870 IF J$(J)="" THEN 3530
1880 X=(J=1)+(J=3)
1890 CALL HCHAR(V+1,4-24*X,145+X,3)
1900 IF POS(J$(J),"@",1)=0 THEN 2350
1910 IF ASC(J$(J))<3 THEN 1990
1920 FOR I=1 TO 8
1930 X$=SEG$(M$(J),I,1)
1940 IF X$="X" THEN 1970
1950 CRTE=ASC(X$)
1960 GOSUB 4630
1970 NEXT I
1980 REM  jeu des joueurs humains
1990 X=0
2000 C=22+(J/2=INT(J/2))*11
2010 C=C+X/4*3
2020 CALL HCHAR(V+3,1,32,32)
2030 CALL SOUND(100,1000,6)
2040 CALL HCHAR(V+3,C,147)
2050 CALL HCHAR(V+3,C+1,148)
2060 FOR I=1 TO 2
2070 CALL KEY(I,K,S)
2080 IF K=18 THEN 2200
2090 CALL JOYST(I,X,S)
2100 IF X THEN 2190
2110 NEXT I
2120 CALL KEY(3,K,S)
2130 IF S=0 THEN 2060
2140 IF K=13 THEN 2200
2150 X=4
2160 IF K=68 THEN 2190
2170 X=-4
2180 IF K<>83 THEN 2060
2190 IF (X=4)*(C>27)+(X=-4)*(C<6)THEN 206
0 ELSE 2010
2200 CALL SOUND(300,-4,4)
2210 CALL HCHAR(V+3,1,32,32)
2220 IF (C<>5)*(C<>28)THEN 2290
2230 IF (M(J,0,5)=8)-P THEN 3430
2240 X=1
2250 GOSUB 4520
2260 GOSUB 4780
2270 P=1
2280 GOTO 1990
2290 X=(J=0)+(J=2)
2300 CRTE=ASC(SEG$(M$(J),ABS((C-5)*X/3+(2
8-C)*(X=0)/3),1))
2310 IF (CRTE=88)+(VL<>11)*(VL<>DVL)*(CL<
>DEM)THEN 1990
2320 DEM=CL
2330 GOTO 3290
2340 REM  jeu et strategie TI
2350 POSS=-2*(M(J,0,DEM)>0)-4*(M(J,0,4)>0
)
2360 CALL SOUND(200,700,5)
2370 FOR I=0 TO 3
2380 IF POS(M$(J),CHR$(I*13+DVL),1)=0 THE
N 2410
2390 POSS=POSS+3
2400 GOTO 2420
2410 NEXT I
2420 IF POSS THEN 2510
2430 IF (M(J,0,5)=8)-P THEN 3430
2440 CALL SOUND(200,250,6)
2450 X=-4+(ASC(J$(J))=2)
2460 GOSUB 4520
2470 X=1
2480 GOSUB 4780
2490 P=1
2500 GOTO 2350
2510 FOR I=1 TO 2
2520 FOR W=0 TO 3
2530 IF M(W,0,5)=I THEN 2580
2540 NEXT W
2550 NEXT I
2560 PERD=0
2570 GOTO 2600
2580 PERD=1
2590 IF M(J,0,5)<=I THEN 2560
2600 IF (POSS>5)*PERD THEN 2740
2610 ON POSS-1 GOTO 2620,2760,2740,3080,2
620,2760,32767,3090
2620 FOR I=10 TO 1 STEP -1
2630 IF I>4 THEN 2680
2640 X=POS(M$(J),CHR$(DEM*13+12),1)
2650 IF X THEN 2710
2660 X=POS(M$(J),CHR$(DEM*13+13),1)
2670 IF X THEN 2710
2680 X=POS(M$(J),CHR$(DEM*13+I),1)
2690 IF X THEN 2710
2700 NEXT I
2710 CRTE=ASC(SEG$(M$(J),X,1))
2720 DEM=CL
2730 GOTO 3290
2740 X=11
2750 GOTO 2770
2760 X=DVL
2770 FOR I=0 TO 3
2780 Y=POS(M$(J),CHR$(I*13+X),1)
2790 IF Y THEN 2810
2800 NEXT I
2810 CRTE=ASC(SEG$(M$(J),Y,1))
2820 IF PERD=0 THEN 3130
2830 K=CRTE
2840 X$=""
2850 FOR X=0 TO 3
2860 IF M(X,0,5)>2 THEN 2920
2870 FOR W=1 TO 8
2880 CRTE=ASC(SEG$(M$(X),W,1))
2890 IF CRTE=88 THEN 2910
2900 X$=X$&STR$(CL)
2910 NEXT W
2920 NEXT X
2930 FOR W=1 TO 8
2940 CRTE=ASC(SEG$(M$(J),W,1))
2950 IF (CRTE=88)-POS(X$,STR$(CL),1)THEN
2980
2960 DEM=CL
2970 GOTO 3060
2980 NEXT W
2990 FOR I=0 TO 3
3000 IF POS(X$,STR$(I),1)THEN 3030
3010 DEM=I
3020 GOTO 3060
3030 NEXT I
3040 CRTE=K
3050 GOTO 3130
3060 CRTE=K
3070 GOTO 3290
3080 IF PERD THEN 2760
3090 FOR K=0 TO 3
```

```
3100 IF POS(M$(J),CHR$(13*K+DVL),1)THEN 3
120
3110 NEXT K
3120 CRTE=K*13+DVL
3130 K=0
3140 FOR C=0 TO 3
3150 X=M(J,0,C)+(CL=C)
3160 IF X<=K THEN 3180
3170 K=X
3180 NEXT C
3190 S=0
3200 FOR C=0 TO 3
3210 X=M(J,1,C)+(CL=C)*PT
3220 IF (M(J,0,C)<K)+(X<S)THEN 3250
3230 W=C
3240 S=X
3250 NEXT C
3260 IF ((W=DEM)*(POSS=5))+((W=DEM)*(POSS
=9))THEN 2620
3270 DEM=W
3280 REM    carte jouee
3290 X$=CHR$(CRTE)
3300 CALL VCHAR(V,H(X$),32,3)
3310 CALL VCHAR(V,H(X$)+1,32,3)
3320 GOSUB 4660
3330 X=POS(M$(J),X$,1)
3340 M$(J)=SEG$(M$(J),1,X-1)&"X"&SEG$(M$(
J),X+1,8)
3350 IF (VL<>11)*(VL<>DVL)-(POS(J$(J),"@"
,1)=0)-(M(J,0,5)=0)THEN 3370
3360 GOSUB 4190
3370 IF DEM<4 THEN 3390
3380 DEM=INT(4*RND)
3390 CALL HCHAR(13,7,110+DEM*8)
3400 DVL=VL
3410 X=-1
3420 GOSUB 4780
3430 X=(J=1)+(J=3)
3440 CALL HCHAR(V+1,4-24*X,32,3)
3450 IF M(J,0,5)=0 THEN 3560
3460 IF ASC(J$(J))<3 THEN 3530
3470 X=-5
3480 FOR I=1 TO 8
3490 X$=SEG$(M$(J),I,1)
3500 IF X$="X" THEN 3520
3510 GOSUB 4630
3520 NEXT I
3530 NEXT J
3540 GOTO 1850
3550 REM   manche gagnee
3560 GOSUB 4880
3570 FOR W=0 TO 3
3580 SC(J)=SC(J)+M(W,1,5)
3590 NEXT W
3600 X$=STR$(SC(J))
3610 X=(J=0)+(J=2)
3620 FOR W=1 TO LEN(X$)
3630 CALL HCHAR(V+2,27+X*24+W,ASC(SEG$(X$
,W,1)))
3640 NEXT W
3650 GOSUB 4110
3660 FOR J=0 TO 3
3670 FOR K=0 TO 1
3680 FOR W=0 TO 5
3690 M(J,K,W)=0
3700 NEXT W
3710 NEXT K
3720 NEXT J
3730 FOR J=0 TO 3
3740 IF SC(J)>59 THEN 3780
3750 NEXT J
3760 GOTO 1560
3770 REM   partie gagnee
3780 J$(J)=J$(J)&"*"
3790 FOR J=0 TO 3
3800 IF J$(J)="" THEN 3870
3810 X$=STR$(VAL(SEG$(J$(J),5,6))+SC(J))
3820 FOR K=LEN(X$)TO 5
3830 X$=" "&X$
3840 NEXT K
3850 J$(J)=SEG$(J$(J),1,4)&X$&SEG$(J$(J),
11,9)
3860 SC(J)=0
3870 NEXT J
3880 CALL CLEAR
3890 CALL COLOR(3,2,1)
3900 CALL COLOR(4,2,1)
3910 FOR J=0 TO 3
3920 IF J$(J)="" THEN 3960
3930 FOR K=1 TO LEN(J$(J))
3940 CALL HCHAR(4+J*3,8+K,ASC(SEG$(J$(J),
K,1)))
3950 NEXT K
3960 NEXT J
3970 GOSUB 4880
3980 GOSUB 4880
3990 FOR K=1 TO 16
4000 CALL HCHAR(20,8+K,ASC(SEG$("M_MEMES
JOUEURS ",K,1)))
4010 CALL HCHAR(22,8+K,ASC(SEG$("A_AUTRES
 JOUEURS",K,1)))
4020 CALL HCHAR(24,8+K,ASC(SEG$("T_TERMIN
ER      ",K,1)))
4030 NEXT K
4040 CALL KEY(3,K,S)
4050 IF K=77 THEN 1400
4060 IF K=65 THEN 780
4070 IF K<>84 THEN 4040
4080 CALL SCREEN(1)
4090 PRINT ::::::::TAB(10);"AU REVOIR"::
::::TAB(9);"ET A BIENTOT":::"    POUR UN
E AUTRE PARTIE."::::::
4100 END
4110 FOR I=0 TO 3
4120 M$(I)=""
4130 FOR W=1 TO 9
4140 M$(I)=M$(I)&"X"
4150 NEXT W
4160 NEXT I
4170 RETURN
4180 REM   choix demande
4190 IF M(J,0,5)=1 THEN 4490
4200 X=0
4210 C=8
4220 CALL HCHAR(13,7,32)
4230 FOR K=1 TO 4
4240 CALL HCHAR(13,2+K*2,102+K*8)
4250 NEXT K
4260 CALL SOUND(300,600,4)
4270 C=C+X/4*2
4280 CALL HCHAR(14,4,32,7)
4290 CALL SOUND(100,250,5)
4300 CALL HCHAR(14,C,146)
4310 FOR I=1 TO 2
4320 CALL KEY(I,K,S)
4330 IF K=18 THEN 4450
4340 CALL JOYST(I,X,S)
4350 IF X THEN 4440
4360 NEXT I
4370 CALL KEY(3,K,S)
4380 IF S=0 THEN 4310
4390 IF K=13 THEN 4450
4400 X=4
4410 IF K=68 THEN 4440
```

```
4420 X=-4
4430 IF K<>83 THEN 4310
4440 IF (X=4)*(C=10)+(X=-4)*(C=4)THEN 431
0 ELSE 4270
4450 CALL SOUND(200,300,8,800,6)
4460 DEM=(C-2)/2-1
4470 CALL HCHAR(13,4,32,7)
4480 CALL HCHAR(14,4,32,7)
4490 RETURN
4500 REM  tirage et affichage
4510 REM       d'une carte
4520 X$=CHR$(INT(52*RND+1))
4530 IF POS(PRI$,X$,1)*JC THEN 4520
4540 IF LEN(PRI$)<46 THEN 4560
4550 PRI$=""
4560 PRI$=PRI$&X$
4570 CRTE=ASC(X$)
4580 IF X=0 THEN 4660
4590 W=POS(M$(J),"X",1)
4600 C=H("X")
4610 M$(J)=SEG$(M$(J),1,W-1)&X$&SEG$(M$(J
),W+1,8)
4620 GOTO 4640
4630 C=H(X$)
4640 L=V
4650 GOTO 4680
4660 L=12
4670 C=14
4680 FOR W=0 TO 1
4690 CALL VCHAR(L,C+W,62-78*(X=-5),3)
4700 NEXT W
4710 IF X=-5 THEN 4760
4720 FOR W=0 TO 1
4730 CALL HCHAR(L+W*2,C+W,110+CL*8)
4740 CALL HCHAR(L+W*2,C+1-W,48+VL)
4750 NEXT W
4760 RETURN
4770 REM  mise a jour de la main
4780 IF VL=11 THEN 4820
4790 M(J,0,CL)=M(J,0,CL)+X
4800 M(J,1,CL)=M(J,1,CL)+X*PT
4810 GOTO 4840
4820 M(J,0,4)=M(J,0,4)+X
4830 M(J,1,4)=M(J,1,4)+X*50
4840 M(J,0,5)=M(J,0,5)+X
4850 M(J,1,5)=M(J,1,5)+X*PT
4860 RETURN
4870 REM  accompagnement musical
4880 CALL SOUND(200,698,5)
4890 CALL SOUND(200,784,5)
4900 CALL SOUND(200,880,5)
4910 CALL SOUND(400,988,5)
4920 CALL SOUND(200,988,5)
4930 CALL SOUND(400,988,5)
4940 CALL SOUND(200,1175,5)
4950 CALL SOUND(500,988,5)
4960 CALL SOUND(500,111,30)
4970 RETURN
```

Le tableur TI-Calc

Sophie Ehster

TI-CALC est un tableur produit par *DON DOUGLAS* (Londres) et destiné à notre TI-99. Il se présente sous la forme d'un module enfichable et peut travailler avec la configuration minimale (console seule et lecteur de cassettes pour sauvegarder les tableaux). Notons que ce logiciel reconnaît l'extension 32K (ce n'est pas le cas du module Gestion de Fichiers) et permet ainsi de travailler sur d'assez grands volumes d'informations. Pour un prix somme toute modique (375 francs à *La Règle à Calcul*), on dispose d'un outil extrêmement puissant qui, par certains aspects, soutient la comparaison avec *Multiplan* (tableur concu par *Microsoft* dont on trouve encore quelques modules épars).

Rappelons brièvement ce qu'est un tableur et quelles sont ses principales applications.

Un tableur est un programme capable de gérer un énorme tableau (96 colonnes et 248 lignes pour TI-CALC) dont on peut définir chaque cellule indépendamment ou relativement à d'autres. Un exemple très simple consiste à définir une case de ce tableau comme égale à la moyenne de toutes les cellules d'une ligne. Les formules qui lient les cellules entre-elles peuvent être extrêmement complexes et faire intervenir des tests.

Le tableur est l'outil de décision par excellence. A partir d'un modèle réel (formules liant diverses variables), on construit un modèle informatique (corrélations liant diverses cellules du tableau) dont on peut étudier le comportement. On peut ainsi tester sa sensibilité par rapport à certaines variables ou extrapoler les formules pour faire du calcul prévisionnel.

Prenons un exemple très concret : un emprunt. Connaissant le montant et la durée d'un emprunt on peut calculer, en fonction du taux d'intérêt, la mensualité à payer et la ventilation de cette somme sur divers postes (intérêts, assurance, capital amorti, etc..).

Aprss avoir décrit dans le tableau les formules liant la durée, le montant de l'emprunt, le taux d'intérêt et la mensualité, on peut faire varier l'un de ces paramètres et apprécier les répercussions sur les autres.

En un tour de main on peut donc analyser avec précision le cas le mieux adapté à telle ou telle situation. TI-CALC se prête très bien à ce type de travail.

Pour les utilisateurs de *Multiplan* ou pour ceux ayant déjà travaillé avec un tableur, citons les principales caractéristiques de celui-ci.

- formats d'affichage et d'impression entièrement définissables par l'utilisateur;
- partage de l'écran en plusieurs fenêtres;
- suppression ou insertion de lignes (ou de colonnes) avec mise à jour automatique des formules;
- blocage à l'écran de certaines lignes (ou colonnes) permettant de visualiser en permanence des titres;
- 4 couleurs disponibles pour chaque cellule;
- panoplie impressionante de fonctions numériques et logiques;
- gestion des erreurs assez évoluée.

Un seul véritable défaut : il n'est pas possible de lier plusieurs tableaux.

Un dernier mot sur la façon dont sont agencés les calculs. Contrairement à de nombreux tableurs, une cellule ne peut pas contenir de formule. Elles ne représentent qu'une grandeur numérique. Tous les calculs sont regroupés dans une liste de calcul faisant référence aux cellules.

Pour conclure, TI-CALC est d'un très bon rapport qualité/prix. Il permet de faire beaucoup plus de choses que les modules *Statistiques* ou *Gestion de Fichiers* et a sa place sur tous les TI soucieux d'épargner de longues séances de calcul à leurs heureux possesseurs. □

Sauvegarde des images graphiques et copie d'écran

Jean-Luc Bazanegue

A en croire le courrier que vous nous envoyez, vous êtes nombreux à utiliser régulièrement les routines graphiques en assembleur. Ceux qui désiraient sauvegarder le résultat de leurs travaux sur disquettes se sont heurtés à un problème : la taille des fichiers de sauvegarde (12Ko) ne permet de stocker que sept images graphiques par disquette. Afin de gagner de la place, nous avons employé la seule solution possible : le codage des images.

Codage des images

Une image graphique en haute résolution est constituée de deux blocs de six Ko pour la table des descripteurs et la table des couleurs. On peut constater, lorsque l'on consulte ces blocs, que les octets consécutifs sont très souvent semblables. Le codage consiste à repérer ces octets identiques pour obtenir une image compactée.

Le codage se fait en plusieurs étapes :

- les deux blocs de mémoire sont subdivisés en 96 blocs de 128 octets. Une fois codé, chacun de ces blocs constituera un enregistrement de taille variable.
- Pour un bloc donné, nous partons du premier octet et nous regardons si l'octet suivant est identique au premier. Si c'est le cas, un compteur est incrémenté et nous recommençons jusqu'à ce que nous trouvions un octet différent du précédent. A ce stade, le bit de poids fort (bit de signe) du compteur est forcé à 1 et l'octet résultant est stocké dans le buffer destiné à l'enregistrement. Nous stockons ensuite la valeur trouvée plusieurs fois de suite.
- Si nous trouvons des octets différents les uns des autres, nous incrémentons aussi un compteur mais, cette fois, en laissant le bit de poids fort à 0. Ainsi, la routine de décodage pourra déterminer si elle a affaire à des octets identiques ou différents. Les octets différents sont stockés à la suite du compteur.
- Nous recommençons l'opération jusqu'au 128ème octet.
- Le deuxième bloc est ensuite traité, et ainsi de suite jusqu'au 96ème.

Pour résumer, nous vous proposons un petit schéma représentant un bloc de 128 octets une fois codé :

Ceci nous permet d'obtenir sur la disquette des fichiers dont la taille est proportionnelle au contenu des images.

La routine de codage et sauvegarde peut être appelée soit depuis le Basic avec **CALL LINK("SAUVE")**, soit depuis la routine "IMAGE" avec la touche **Contrôle-S**.

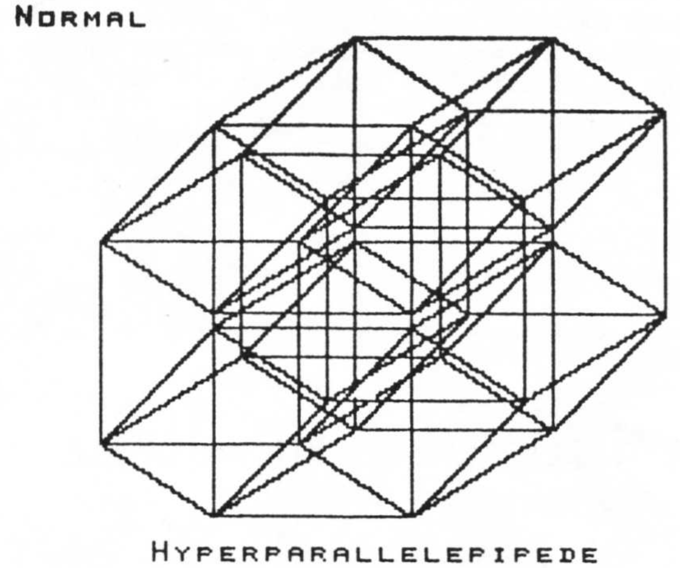
La routine de chargement et décodage peut être appelée depuis le Basic avec **CALL LINK("CHARGE")** ou depuis la routine "IMAGE" avec **Contrôle-C**.
Dans les deux cas, la disquette de sauvegarde doit se trouver dans le lecteur 1 et il est possible de retourner au Basic avec la touche **BACK**.

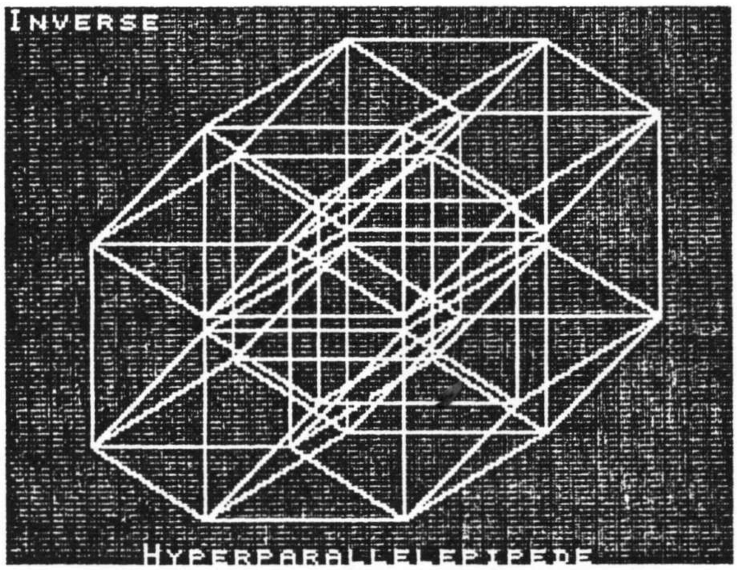
>A3	Le poids fort du compteur est à 1 donc l'octet >FF
>FF	doit être répété 35 fois (>A3=163, 163-128=35).
>05	Le poids fort du compteur est à 0 donc les 5 octets
>67	qui suivent doivent être transmis normalement.
>8A	
>21	35 + 5 + 88 = 128 octets
>AA	
>F2	
>D8	Le poids fort du compteur est à 1 donc l'octet >00
>00	doit être répété 88 fois (>D8=216, 216-128-88).

Copie d'écran

En plus des procédures de sauvegarde et de chargement, nous avons ajouté aux routines graphiques un programme de copie d'écran sur imprimante Apple ImageWriter (liaison série). Notre choix s'est porté sur cette imprimante en raison de son excellent rapport qualité/prix. Comme tous nos lecteurs ne disposent pas de ce type d'imprimante, nous avons placé dans le listing des indications qui vous permettront d'adapter la routine à votre imprimante. Sur la disquette "Routines graphiques", nous avons enregistré, en plus du code objet contenant la copie d'écran sur ImageWriter (OBJETI), une version qui fonctionne avec une imprimante Apple DMP (liaison parallèle) sous le nom "OBJETD". Ces codes objets fonctionnent indifféremment avec l'éditeur/assembleur ou la Mini-mémoire. Par la suite, nous ajouterons les codes objets pour d'autres imprimantes (GP100, Epson, etc...).

Pour obtenir une copie d'écran, lorsque vous êtes dans la routine "IMAGE", il suffit d'appuyer sur **CTRL-N** pour obtenir une copie normale, ou **CTRL-I** pour obtenir une copie en inverse.





Les routines graphiques ont été plusieurs fois modifiées depuis leurs débuts dans le numéro 4, il nous semble donc indispensable de vous donner le listing complet du source. De plus, ceux d'entre-vous qui disposent seulement de la Mini-mémoire nous ont fait part de leur désir de voir publié un programme Basic contenant le code objet, comme dans le numéro 4. C'est désormais chose faite. Ce programme contient toutes les routines sauf la sauvegarde des images, inutile si vous n'avez pas de lecteur de disquettes. □

```
       DEF   EFFT,EFFP,EFFC,AXES
       DEF   POINT,IMAGE,LIGNE
       DEF   CHARGE,SAUVE,LETTRE

       REF   VMBW,DSRLNK,VSBW
       REF   VMBR,VSBR,KSCAN,VWTR

       AORG >A000

COORD  DATA 0
COULP  DATA 0
*
COULE  DATA 0
POSAXE DATA 0
LIGNED DATA 0
LIGNEA DATA 0
TYPEL  BYTE 0
CODEL  BYTE 0
LIGNEL BYTE 0
COLONL BYTE 0
CONL   BYTE 0
COFFL  BYTE 0
REG11  DATA 0
```

```
CR7    DATA 0
CR8    DATA 0
SARGPL DATA 0

RVDP1  DATA >E181
RVDP1T DATA >F081

REGVDP DATA >0280,>A181,>0682,>FF83
       DATA >0384,>3685,>0786,>1187,0
RVDPBA DATA >0080,>E081,>0082,>0C83
       DATA >0084,>0685,>0086,>0787,0
RVDPT  DATA >0080,>B081,>0082,>0E83
       DATA >0084,>0685,>0086,>F487,0

**************************************
* IMP = parametres pour fichier      *
* Imprimante ImageWriter (serie)     *
* >0012 = OPEN, VARIABLE, DISPLAY,   *
*         OUTPUT, SEQUENTIAL         *
* >1000 = Adresse du buffer en       *
*         video                      *
* >FE00 = Maximum 254 octets         *
* >0017 = longueur nom du periph.    *
```

```
***************************************
* Pour une imprimante DMP             *
* (parallele), remplacer ce qui suit  *
* par :                               *
* IMP   DATA >0012,>1'000,>FF00       *
*       DATA >0000,>0003              *
*       TEXT 'PIO'                    *
*       EVEN                          *
***************************************
*                                     *
IMP     DATA >0012,>1000,>FF00        *
        DATA >0000,>0017              *
        TEXT 'RS232.BA=9600.DA=8.PA=N' *
        EVEN                          *
*                                     *
***************************************

***************************************
* DSK = Parametres pour fichier       *
* lecteur de disquettes :             *
* >001A = OPEN, VARIABLE, INTERNAL,   *.
*         OUTPUT, SEQUENTIAL          *
***************************************

DSK     DATA >001A,>1000,>FE00
        DATA >0000,>0000
TDSK    TEXT 'DSK1.

***************************************
* Codes ASCII + 96 des messages pour  *
* les routines "CHARGE" et "SAUVE"    *
***************************************

TCHARG BYTE 163,200,193,210,199
       BYTE 197,205,197,206,212
TSAUVE BYTE 179,193,213,214,197
       BYTE 199,193,210,196,197
TERR1  BYTE 165,210,210,197,213,210
       BYTE 128,195,207,196,197
TERR2  BYTE 156,165,174,180,165,178,158
       BYTE 128,208,207,213,210,128,195
       BYTE 207,206,212,201,206,213,197
       BYTE 210,142,142,142

TCOUL  EQU  >2676
ETAT   EQU  >837C
RCLAV  EQU  >8375
PLONG  EQU  >8356
TPAB   EQU  >1000
PAB    EQU  >F80
BACK   BYTE 15
CN     BYTE 142
CI     BYTE 137
CS     BYTE 147
CC     BYTE 131
TENT   BYTE >0D
TFS    BYTE >08
TFD    BYTE >09
CURS   BYTE 111
CMINI  BYTE >30
CMAXI  BYTE >5A

***************************************
* Caracteres de controle pour         *
* la copie d'ecran :                  *
*                                     *
* LF8 = reglage du "LINE FEED" et     *
*       impression unidirectionnelle  *
* ( CHR$(27);"c";CHR$(27);"n";        *
* CHR$(27);"T16" )                    *
*                                     *
* GRAPH = Mode graphique et nombre    *
*         d'octets transmis           *
* ( CHR$(27);"G0192" )                *
*                                     *
* RESET = reinitialisation de         *
*         l'imprimante                *
* ( CHR$(27);"c" )                    *
***************************************

LF8     DATA >1B3E,>1B6E,>1B54,>3136
GRAPH   DATA >1B47,>3031,>3932
RESET   DATA >1B63
        EVEN

LIGNE   MOVB @LIGNED,R9
        SRL  R9,8
        MOVB @LIGNEA,R7
        SRL  R7,8
        MOV  R7,@CR7
        MOVB @LIGNED+1,R2
        SRL  R2,8
        CI   R2,191
        JGT  LIGNE8
        LI   R10,191
        S    R2,R10
        MOVB @LIGNEA+1,R2
        SRL  R2,8
        CI   R2,191
        JGT  LIGNE8
        LI   R8,191
        S    R2,R8
        MOV  R8,@CR8

        CLR  R12
        LI   R5,1
        LI   R6,1
        S    R9,R7
        JLT  LIGNE1
        JMP  LIGNE2
LIGNE1  NEG  R5
        NEG  R7
LIGNE2  MOV  R7,R7
        JNE  LIGNE3
        SETO R12
LIGNE3  S    R10,R8
        JLT  LIGNE4
        JMP  LIGNE5
LIGNE4  NEG  R6
        NEG  R8
LIGNE5  MOV  R9,R0
        MOV  R10,R1
        MOV  R1,R4
        SLA  R4,5
        SOC  R1,R4
        ANDI R4,>FF07
        A    R0,R4
        ANDI R0,7
        S    R0,R4
        LI   R3,>8000
        SRC  R3,0
        SOCB R3,@TIMAGE(R4)
        MOVB @COULP,R13
        CLR  R2
        MOVB @COULP+1,R2
        SRC  R2,12
        SOCB R2,R13
        MOVB R13,@TCOUL(R4)
        C    R9,@CR7
        JNE  LIGNE6
        C    R10,@CR8
        JNE  LIGNE6
LIGNE8  B    @RETOUR
LIGNE6  MOV  R12,R12
```

```
        JLT   LIGNE7
        A     R5,R9
        S     R8,R12
        JMP   LIGNE5
LIGNE7  A     R6,R10
        A     R7,R12
        JMP   LIGNE5

EFFT    SETO  R2
        JMP   EFFP2
EFFP    CLR   R2
EFFP2   LI    R0,TIMAGE
EFFP1   CLR   *R0+
        CI    R0,TIMAGE+>1800
        JNE   EFFP1
        MOV   R2,R2
        JEQ   FEFF
EFFC    LI    R0,TCOUL
EFFC1   CLR   *R0+
        CI    R0,TCOUL+>1800
        JNE   EFFC1
FEFF    B     @RETOUR

AXES    MOV   R11,@REG11

        CLR   R8
        MOVB  @POSAXE+1,R12
        SRL   R12,8
        CI    R12,191
        JGT   AXEY1
        LI    R9,191
        S     R12,R9
        JMP   AXEY
AXEY1   CLR   R9
AXEY    MOV   R8,R0
        MOV   R9,R1
        BL    @POINT2
        INC   R8
        CI    R8,256
        JNE   AXEY

        CLR   R9
        MOVB  @POSAXE,R8
        SRL   R8,8
AXEX    MOV   R8,R0
        MOV   R9,R1
        BL    @POINT2
        INC   R9
        CI    R9,192
        JNE   AXEX
        MOV   @REG11,R11
        B     @RETOUR

POINT   SETO  R7
        MOVB  @COORD,R0
        SRL   R0,8
        MOVB  @COORD+1,R2
        SRL   R2,8
        CI    R2,191
        JGT   FHPLOT
        LI    R1,191
        S     R2,R1
POINT2  CLR   R7
        MOV   R1,R4
        SLA   R4,5
        SOC   R1,R4
        ANDI  R4,>FF07
        A     R0,R4
        ANDI  R0,7
        S     R0,R4
        LI    R3,>8000
        SRC   R3,0
```

```
        SOCB  R3,@TIMAGE(R4)
        MOVB  @COULP,R5
        CLR   R6
        MOVB  @COULP+1,R6
        SRC   R6,12
        SOCB  R6,R5
        MOVB  R5,@TCOUL(R4)
        MOV   R7,R7
        JNE   FHPLOT
        RT
FHPLOT  B     @RETOUR

IMAGE   LI    R1,TBASIC
        CLR   R0
        LI    R2,TBASIC+>3800
        MOVB  R0,@>8C02
        NOP
        MOVB  R0,@>8C02
        NOP
SB      MOVB  @>8800,*R1+
        C     R1,R2
        JNE   SB

IMAGE2  MOVB  @COULE,@REGVDP+14

        MOVB  @REGVDP+2,@>83D4
        LI    R1,REGVDP
IRBM    MOV   *R1+,R2
        JEQ   ITLUT
        MOVB  R2,@>8C02
        SWPB  R2
        MOVB  R2,@>8C02
        JMP   IRBM

ITLUT   LI    R0,>005B
        LI    R1,>D000
        MOVB  R0,@>8C02
        SWPB  R0
        MOVB  R0,@>8C02
        NOP
        MOVB  R1,@>8C00

        CLR   R1
        LI    R2,768
        LI    R0,>0058
        MOVB  R0,@>8C02
        SWPB  R0
        MOVB  R0,@>8C02
ITE     SWPB  R1
        MOVB  R1,@>8C00
        SWPB  R1
        INC   R1
        C     R1,R2
        JNE   ITE

        LI    R0,>0040
        LI    R1,TIMAGE
        LI    R2,TIMAGE+>1800
        MOVB  R0,@>8C02
        SWPB  R0
        MOVB  R0,@>8C02
        NOP
TTI     MOVB  *R1+,@>8C00
        C     R1,R2
        JNE   TTI

        LI    R0,>0060
        LI    R1,TCOUL
        LI    R2,TCOUL+>1800
        MOVB  R0,@>8C02
        SWPB  R0
        MOVB  R0,@>8C02
```

```
       NOP
TTC    MOVB *R1+,@>8C00
       C    R1,R2
       JNE  TTC

       MOVB @RVDP1,@>83D4
       MOV  @RVDP1,R0
       MOVB R0,@>8C02
       SWPB R0
       MOVB R0,@>8C02

CLAV1  LWPI >83E0
CLAV   BL   @>000E
       LWPI >20BA
       CB   @RCLAV,@BACK
       JEQ  RBIS
       CB   @RCLAV,@CN
       JNE  CLAV0
       B    @IMPN
CLAV0  CB   @RCLAV,@CI
       JNE  CLAV2
       B    @IMPI
CLAV2  CB   @RCLAV,@CS
       JNE  CLAV3
       CLR  R10
CLAV4  B    @SETC
CLAV3  CB   @RCLAV,@CC
       JNE  CLAV1
       SETO R10
       JMP  CLAV4

RBIS   LI   R1,RVDPBA
IRB    MOV  *R1+,R2
       JEQ  TTBAS
       MOVB R2,@>8C02
       SWPB R2
       MOVB R2,@>8C02
       JMP  IRB

TTBAS  LI   R0,>0040
       LI   R1,TBASIC
       LI   R2,TBASIC+>3800
       MOVB R0,@>8C02
       SWPB R0
       MOVB R0,@>8C02
       NOP
TTB    MOVB *R1+,@>8C00
       C    R1,R2
       JNE  TTB

RETOUR CLR  R0
       MOVB R0,@ETAT
       RT

LETTRE MOVB @CODEL,R14
       SRL  R14,8
       CI   R14,32
       JH   L0
L3     B    @RETOUR
L0     MOVB @TYPEL,R15
       JEQ  L1
       LI   R13,2304
       CI   R14,91
       JL   L2
       JMP  L3
L1     LI   R13,1024
       CI   R14,159
       JH   L3
L2     AI   R14,-32
       SLA  R14,3
       A    R14,R13
       MOVB @COLONL,R14
       SRL  R14,8
       CI   R14,32
       JH   L3
       DEC  R14
       JNC  L3
       SLA  R14,3
       MOVB @LIGNEL,R12
       SRL  R12,8
       CI   R12,24
       JH   L3
       DEC  R12
       JNC  L3
       MOV  R12,R12
       JEQ  L4
L5     AI   R14,256
       DEC  R12
       JNE  L5
L4     MOVB @COFFL,R9
       CLR  R10
       MOVB @CONL,R10
       SRC  R10,12
       SOC  R10,R9
       LI   R0,TCOUL
       A    R14,R0
       LI   R1,8
L6     MOVB R9,*R0+
       DEC  R1
       JNE  L6
       LI   R0,TIMAGE
       A    R14,R0
       LI   R1,8
       SWPB R13
       MOVB R13,@>8C02
       SWPB R13
       MOVB R13,@>8C02
       NOP
L7     MOVB @>8800,*R0+
       DEC  R1
       JNE  L7
       JMP  L3

**************************************
* Routine de copie de l'image        *
* graphique sur imprimante           *
* ImageWriter Apple                  *
*                                    *
* CTRL-N pour impression normale     *
* CTRL-I pour impression en inverse *
**************************************

IMPI   SETO R10
       JMP  IMP0
IMPN   CLR  R10

*******************************
* Sauvegarde de l'adresse de *
* retour GPL                  *
*******************************
* Pas necessaire avec         *
* l'imprimante DMP            *
*******************************
*                             *
IMP0   MOVB @>9802,@SARGPL    *
       MOVB @>9802,@SARGPL+1 *
       DEC  @SARGPL           *
*                             *
*******************************

* Ouverture du fichier

       LI   R0,PAB
       LI   R1,IMP
```

```
       LI    R2,40
       BLWP  @VMBW
       LI    R9,PAB+9
       MOV   R9,@PLONG
       BLWP  @DSRLNK
       DATA  8

* Initialisation du bloc d'acces
* aux peripheriques pour ecriture

       LI    R0,PAB
       LI    R1,>0300
       BLWP  @VSBW

* transmission des codes pour
* line feed et impression
* unidirectionnelle

       LI    R0,TPAB
       LI    R1,LF8
       LI    R2,8
       BLWP  @VMBW
       LI    R0,PAB+5
       LI    R1,>0800
       BLWP  @VSBW
       LI    R9,PAB+9
       MOV   R9,@PLONG
       BLWP  @DSRLNK
       DATA  8

**********************
* Boucle d'impression *
**********************

BIMP   LI    R3,32
       LI    R4,TIMAGE+248

BIMP0  MOV   R4,R5

* Transmission des codes pour
* le mode graphique et le
* nombre d'octets

BIMP1  LI    R0,TPAB
       LI    R1,GRAPH
       LI    R2,6
       BLWP  @VMBW
       LI    R0,TPAB+6
       LI    R6,192

* Calcul des codes a transmettre

BIMP2  LI    R7,8

BIMP5  MOVB  *R5,R1
       MOV   R10,R10
       JEQ   BIMP3
       INV   R1
BIMP3  BLWP  @VSBW
       DEC   R6
       JEQ   IMPR
       INC   R0
       DEC   R7
       JEQ   BIMP4
       INC   R5
       JMP   BIMP5
BIMP4  AI    R5,249
       JMP   BIMP2

* Impression d'une ligne

IMPR   LI    R0,PAB+5
       LI    R1,>C600
       BLWP  @VSBW
       LI    R9,PAB+9
       MOV   R9,@PLONG
       BLWP  @DSRLNK
       DATA  8

       DEC   R3
       JEQ   FIMP
       AI    R4,-8
       JMP   BIMP0

************************
* Reinitialisation de  *
* l'imprimante         *
************************
* Ces codes ne sont pas *
* reconnus par         *
* l'imprimante DMP mais *
* la routine peut      *
* rester dans le       *
* programme            *
************************
*                      *
FIMP   LI    R0,TPAB   *
       LI    R1,RESET  *
       LI    R2,2      *
       BLWP  @VMBW     *
       LI    R0,PAB+5  *
       LI    R1,>0200  *
       BLWP  @VSBW     *
       MOV   R9,@PLONG *
       BLWP  @DSRLNK   *
       DATA  8         *
*                      *
************************

* Fermeture du fichier

       LI    R0,PAB
       LI    R1,>0100
       BLWP  @VSBW
       MOV   R9,@PLONG
       BLWP  @DSRLNK
       DATA  8

* Remise en etat de
* l'image graphique

       LI    R0,PAB
       LI    R1,TIMAGE+>0F80
       LI    R2,512
       BLWP  @VMBW

*******************************
* Recuperation de l'adresse de *
* Retour GPL                  *
*******************************
* Pas necessaire avec         *
* l'imprimante DMP            *
*******************************
*                             *
       MOVB @SARGPL,@>9C02    *
       MOVB @SARGPL+1,@>9C02  *
*                             *
*******************************

       B     @CLAV1

*************************************
* Routines de codage, sauvegarde,   *
* chargement et decodage des images *
```

```
* graphiques.                          *
* Appel des routines depuis le Basic *
* par CALL LINK("CHARGE") et           *
* CALL LINK("SAUVE") ou, depuis la     *
* routine "IMAGE", par CTRL-C et       *
* CTRL-S.                              *
****************************************

CHARGE SETO R10
       JMP  C0
SAUVE  CLR  R10

C0     MOVB @RVDPT+2,@>83D4
       MOV  @RVDPT+2,R0
       MOVB R0,@>8C02
       SWPB R0
       MOVB R0,@>8C02

* Sauvegarde de la memoire video
* si appel depuis le Basic.

       CLR  R0
       LI   R1,TBASIC
       LI   R2,TBASIC+>3800
       MOVB R0,@>8C02
       NOP
       MOVB R0,@>8C02
       NOP
C8     MOVB @>8800,*R1+
       C    R1,R2
       JNE  C8
       JMP  C9

* Restitution de la memoire video
* si appel depuis "IMAGE"

SETC   MOVB @RVDPT+2,@>83D4
       MOV  @RVDPT+2,R0
       MOVB R0,@>8C02
       SWPB R0
       MOVB R0,@>8C02
       LI   R0,>0040
       LI   R1,TBASIC
       LI   R2,TBASIC+>3800
       MOVB R0,@>8C02
       SWPB R0
       MOVB R0,@>8C02
       NOP
C10    MOVB *R1+,@>8C00
       C    R1,R2
       JNE  C10

* Passage en mode texte

C9     LI   R1,RVDPT
C1     MOV  *R1+,R2
       JEQ  C2
       MOVB R2,@>8C02
       SWPB R2
       MOVB R2,@>8C02
       JMP  C1

* Efface l'affichage

C2     LI   R2,960
       LI   R0,>0040
       LI   R1,>8000
       MOVB R0,@>8C02
       SWPB R0
       MOVB R0,@>8C02
       NOP
C3     MOVB R1,@>8C00
       DEC  R2
       JNE  C3

* Retablissement de l'affichage

       MOVB @RVDP1T,@>83D4
       MOV  @RVDP1T,R0
       MOVB R0,@>8C02
       SWPB R0
       MOVB R0,@>8C02

* Affichage

       LI   R0,2
       LI   R2,10
       MOV  R10,R10
       JEQ  C4
       LI   R1,TCHARG
       JMP  C5
C4     LI   R1,TSAUVE
C5     BLWP @VMBW

C7     LI   R0,82
       LI   R3,TDSK
       LI   R2,16
C6     MOVB *R3+,R1
       AI   R1,>6000
       BLWP @VSBW
       INC  R0
       DEC  R2
       JNE  C6

*************************
* Saisie nom de fichier *
*************************

       LI   R0,87
       CLR  R7
       LI   R3,>AAAA
       LI   R5,TDSK+5
CB1    LI   R4,>100
CB2    BLWP @KSCAN
       MOVB @ETAT,R6
       JEQ  CBCL
       CB   @RCLAV,@BACK
       JNE  CB3
CBR    LI   R0,>E000
       MOVB R0,@>83D4
       B    @RBIS
CB3    CB   @RCLAV,@TENT
       JEQ  CBENT
       CB   @RCLAV,@TFS
       JEQ  CBFS
       CB   @RCLAV,@TFD
       JEQ  CBFD

* Caracteres

       CI   R7,10
       JEQ  CBCL
       CB   @RCLAV,@CMINI
       JL   CBCL
       CB   @RCLAV,@CMAXI
       JH   CBCL
       MOVB @RCLAV,*R5
       MOVB *R5+,R1
       AI   R1,>6000
       BLWP @VSBW
       INC  R0
       INC  R7
       JMP  CBCL
```

```
* FCTN-S

CBFS    MOV   R7,R7
        JEQ   CBCL
        MOVB  *R5,R1
        AI    R1,>6000
        BLWP  @VSBW
        DEC   R0
        DEC   R7
        DEC   R5
        JMP   CBCL

* FCTN-D

CBFD    CI    R7,10
        JEQ   CBCL
        MOVB  *R5+,R1
        AI    R1,>6000
        BLWP  @VSBW
        INC   R0
        INC   R7

* Clignotement curseur

CBCL    DEC   R4
        JNE   CB2
        SRC   R3,1
        JOC   CBCL1
        MOVB  *R5,R1
        JMP   CBCL2
CBCL1   MOVB  @CURS,R1
CBCL2   AI    R1,>6000
        BLWP  @VSBW
        JMP   CB1

* <ENTER>

CBENT   MOV   R7,R7
        JEQ   CBCL
        CI    R7,10
        JEQ   CBENT3
        LI    R0,TDSK+14
        LI    R2,10
        LI    R3,>2000
CBENT1  MOVB  R3,*R0
        DEC   R0
        DEC   R2
        C     R2,R7
        JNE   CBENT1

CBENT3  LI    R0,82
        LI    R2,16
        LI    R3,TDSK
CBENT2  MOVB  *R3+,R1
        AI    R1,>6000
        BLWP  @VSBW
        INC   R0
        DEC   R2
        JNE   CBENT2

* Ouverture du fichier

        LI    R0,PAB
        LI    R1,DSK
        LI    R2,25
        BLWP  @VMBW
        AI    R7,5
        SLA   R7,8
        MOV   R7,R1
        LI    R0,PAB+9
        BLWP  @VSBW
        LI    R0,PAB+1
        MOV   R10,R10
        JEQ   SC1
        LI    R1,>1C00
        JMP   SC2
SC1     LI    R1,>1A00
SC2     BLWP  @VSBW
        LI    R0,PAB+9
        MOV   R0,@PLONG
        CLR   R1
        MOVB  R1,@ETAT
        BLWP  @DSRLNK
        DATA  8
        JNE   SC

* Erreurs Entree/Sortie

BERR2   LI    R0,PAB+1
        CLR   R1
        BLWP  @VSBR
        ANDI  R1,>E000
        MOV   R1,R3
        LI    R0,162
        LI    R1,TERR1
        LI    R2,11
        BLWP  @VMBW
        LI    R0,202
        LI    R1,TERR2
        LI    R2,25
        BLWP  @VMBW
        LI    R0,174
        LI    R1,144
        SRL   R3,13
        A     R3,R1
        SWPB  R1
        BLWP  @VSBW

BERR    BLWP  @KSCAN
        MOVB  @ETAT,R1
        JEQ   BERR
        CB    @RCLAV,@TENT
        JNE   BERR
        LI    R0,162
        LI    R2,227
        LI    R1,>8000
BERR1   BLWP  @VSBW
        INC   R0
        C     R0,R2
        JNE   BERR1
        B     @C7

SC      MOV   R11,@REG11
        MOV   R10,R10
        JNE   CHAR

************************
* Codage et sauvegarde *
************************

        LI    R1,>0300
        LI    R0,PAB
        BLWP  @VSBW

        BL    @COD
        DATA  TIMAGE

        BL    @COD
        DATA  TCOUL

* Enregistrement couleur ecran

        MOVB  @COULE,R1
        LI    R0,TPAB
```

```
        BLWP @VSBW
        LI   R1,>0100
        LI   R0,PAB+5
        BLWP @VSBW
        LI   R9,PAB+9
        MOV  R9,@PLONG
        BLWP @DSRLNK
        DATA 8
        JNE  SC0
        B    @ENRC3

* Fermeture du fichier et
* retour au Basic

SC0     LI   R1,>0100
        LI   R0,PAB
        BLWP @VSBW
        MOV  R9,@PLONG
        BLWP @DSRLNK
        DATA 8
        MOV  @REG11,R11
        B    @CBR

*************************
* Chargement et decodage *
*************************

CHAR    LI   R1,>0200
        LI   R0,PAB
        BLWP @VSBW

        BL   @DCOD
        DATA TIMAGE

        BL   @DCOD
        DATA TCOUL

* Lecture couleur ecran

        LI   R9,PAB+9
        MOV  R9,@PLONG
        BLWP @DSRLNK
        DATA 8
        JNE  CHAR0
        B    @DCOD4

CHAR0   LI   R0,TPAB
        BLWP @VSBR
        MOVB R1,@COULE

* Fermeture du fichier et
* branchement sur la
* routine "IMAGE"

        LI   R1,>0100
        LI   R0,PAB
        BLWP @VSBW
        MOV  R9,@PLONG
        BLWP @DSRLNK
        DATA 8
        MOV  @REG11,R11
        B    @IMAGE2

********************
* Routine de codage *
********************

COD     MOV  *R11+,R3
        LI   R4,48

COD3    CLR  R5
        MOV  R3,R7
        CLR  R9

COD0    CI   R9,127
        JEQ  COD4
        MOVB *R7+,R8
        CB   *R7,R8
        JNE  CODD

* Octets identiques

        DEC  R7
        CLR  R6
COD1    MOVB *R7+,R8
        INC  R9
        CI   R9,128
        JEQ  COD2
        CB   *R7,R8
        JNE  COD2
        INC  R6
        JMP  COD1

COD2    SWPB R6
        ORI  R6,>8000
        MOVB R6,@TRAM(R5)
        INC  R5
        DEC  R7
        MOVB *R7+,@TRAM(R5)
        INC  R5
        CI   R9,128
        JEQ  ENRC
        JMP  COD0

COD4    CLR  R6
        MOV  R6,@TRAM(R5)
        INC  R5
        MOV  *R7,@TRAM(R5)
        INC  R5
        JMP  ENRC

* Octets differents

CODD    DEC  R7
        MOV  R7,R10
        CLR  R6
CODD0   MOVB *R7+,R8
        INC  R9
        CI   R9,128
        JEQ  CODD1
        CB   *R7,R8
        JEQ  CODD1
        INC  R6
        JMP  CODD0

CODD1   MOV  R6,R2
        SWPB R6
        MOVB R6,@TRAM(R5)
        INC  R5

CODD2   MOVB *R10+,@TRAM(R5)
        INC  R5
        DEC  R2
        JOC  CODD2
        CI   R9,128
        JNE  COD0

* Enregistrement d'un bloc
* de 128 octets code

ENRC    MOV  R5,R2
        SLA  R5,8
        MOVB R5,R1
        LI   R0,PAB+5
```

```
       BLWP @VSBW
       LI   R0,TPAB
       LI   R1,TRAM
       BLWP @VMBW
       LI   R0,PAB+9
       MOV  R0,@PLONG
       BLWP @DSRLNK
       DATA 8
       JNE  ENRC1
ENRC3  CLR  R10
ENRC2  MOV  @REG11,R11
       B    @BERR2
ENRC1  DEC  R4
       JNE  ENRC0
       RT
ENRC0  AI   R3,128
       JMP  COD3
***********************
* Routine de decodage *
***********************
DCOD   MOV  *R11+,R3
       LI   R4,48
DCOD1  MOV  R3,R7
       LI   R0,PAB+9
       MOV  R0,@PLONG
       BLWP @DSRLNK
       DATA 8
       JNE  DCOD0
DCOD4  SETO R10
       JMP  ENRC2

* Lecture des codes
* relatifs a un bloc
* de 128 octets

DCOD0  LI   R0,PAB+5
       BLWP @VSBR
       SRL  R1,8
       MOV  R1,R2
       LI   R0,TPAB
       LI   R1,TRAM
       BLWP @VMBR

       CLR  R5
       CLR  R9

DCOD3  MOVB @TRAM(R5),R8
       ANDI R8,>8000
       JEQ  DCODD

* Octets identiques

       MOVB @TRAM(R5),R8
       INC  R5
       SWPB R8
       ANDI R8,>007F
DCOD2  MOVB @TRAM(R5),*R7+
       INC  R9
       CI   R9,128
       JEQ  FDCOD
       DEC  R8
       JOC  DCOD2

       INC  R5
       JMP  DCOD3
* Octets differents

DCODD  MOVB @TRAM(R5),R8
       INC  R5
       SRL  R8,8
DCODD0 MOVB @TRAM(R5),*R7+
       INC  R9
       CI   R9,128
       JEQ  FDCOD
       INC  R5
       DEC  R8
       JOC  DCODD0
       JMP  DCOD3

FDCOD  DEC  R4
       JNE  FDCOD0
       RT

FDCOD0 AI   R3,128
       JMP  DCOD1

TBASIC BSS  >3800
TIMAGE BSS  >1800
TRAM   BSS  254

       END
```

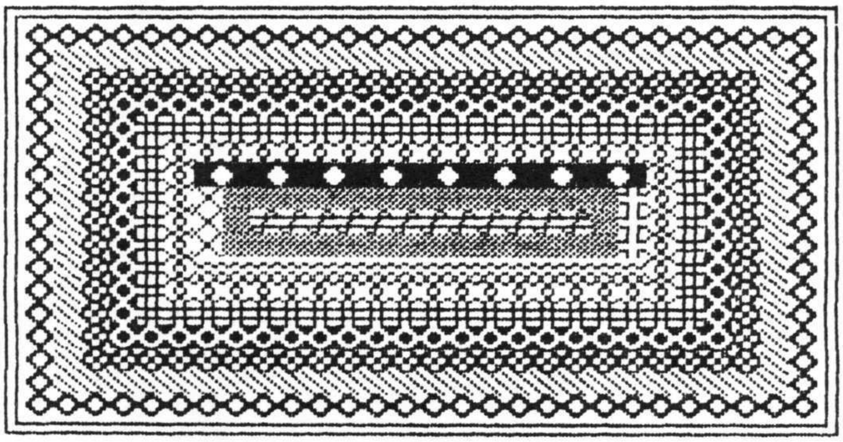
```
100 REM  ********************
110 REM  *     Routines     *
120 REM  *    Graphiques    *
130 REM  ********************
140 REM  *   Ce programme   *
150 REM  * implante le code *
160 REM  * objet en memoire *
170 REM  ********************
180 REM  *    Basic TI +    *
190 REM  *  Mini-memoire +  *
200 REM  *  Extension 32Ko  *
210 REM  ********************
220 REM
230 REM  Pointeur derniere
240 REM  adresse libre dans
250 REM  la Mini-memoire
260 REM
270 CALL LOAD(28702,127,192)
280 REM
290 REM  Initialisation de
300 REM  la table des
310 REM  references externes
320 REM
330 DATA 69,70,70,84,32,32,161,146,69,70,
70,80,32,32,161,150,69,70,70,67,32,32,1
61,168
340 DATA 65,88,69,83,32,32,161,184,80,79,
73,78,84,32,162,4,73,77,65,71,69,32,162
,88
350 DATA 76,73,71,78,69,32,160,228,76,69,
84,84,82,69,163,166
360 FOR I=32704 TO 32767
370 READ N
380 CALL LOAD(I,N)
390 NEXT I
400 REM
410 REM  Chargement du code
420 REM  objet dans
430 REM  l'extension 32Ko
440 REM
450 DATA 225,129,240,129,2,128,161,129,6,
130,255,131,3,132,54,133,7,134,17,135,0
,0
460 DATA 0,128,224,129,0,130,12,131,0,132
,6,133,0,134,7,135,0,0,0,128,176,129,0,
130
470 DATA 14,131,0,132,6,133,0,134,244,135
,0,0
480 DATA 0,18,16,0,255,0,0,0,0,23,82,83,5
0,51,50,46,66,65,61,57,54,48,48,46,68,6
5,61
490 DATA 56,46,80,65,61,78,0,0,26,16,0,25
4,0,0,0,0,0,68,83,75,49,46
```

```
500 DATA 32,32,32,32,32,32,32,32,32,32,32
,163,200,193,210,199,197,205,197,206,21
2
510 DATA 179,193,213,214,197,199,193,210,
196,197,165,210,210,197,213,210,128,195
,207
520 DATA 196,197,156,165,174,180,165,178,
158,128,208,207,213,210,128,195,207,206
,212
530 DATA 201,206,213,197,210,142,142,142,
15,142,137,147,131,13,8,9,111,48,90,0,2
7,62
540 DATA 27,110,27,84,49,54,27,71,48,49,5
7,50,27,99,210,96,160,8,9,137,209,224,1
60,10
550 DATA 9,135,200,7,160,20,208,160,160,9
,9,130,2,130,0,191,21,63,2,10,0,191,98,
130
560 DATA 208,160,160,11,9,130,2,130,0,191
,21,54,2,8,0,191,98,2,200,8,160,22,4,20
4
570 DATA 2,5,0,1,2,6,0,1,97,201,17,1,16,2
,5,5,5,7,193,199,22,1,7,12,98,10,17,1,1
6,2
580 DATA 5,6,5,8,192,9,192,74,193,1,10,84
,225,1,2,68,255,7,161,0,2,64,0,7,97,0,2
,3
590 DATA 128,0,11,3,249,3,226,38,211,96,1
60,2,4,194,208,160,160,3,11,194,243,66
600 DATA 217,13,38,118,136,9,160,20,22,5,
136,10,160,22,22,2,4,96,163,158,195,12
610 DATA 17,3,162,69,99,8,16,218,162,134,
163,7,16,215,7,2,16,1,4,194,2,0,226,38
620 DATA 4,240,2,128,250,38,22,252,192,13
0,19,6,2,0,38,118,4,240,2,128,62,118,22
,252
630 DATA 4,96,163,158,200,11,160,18,4,200
,211,32,160,7,9,140,2,140,0,191,21,4,2,
9
640 DATA 0,191,98,76,16,1,4,201,192,8,192
,73,6,160,162,30,5,136,2,136,1,0,22,248
650 DATA 4,201,210,32,160,6,9,136,192,8,1
92,73,6,160,162,30,5,137,2,137,0,192,22
,248
660 DATA 194,224,160,18,4,96,163,158,7,7,
208,32,160,0,9,128,208,160,160,1,9,130
670 DATA 2,130,0,191,21,30,2,1,0,191,96,6
6,4,199,193,1,10,84,225,1,2,68,255,7,16
1,0
680 DATA 2,64,0,7,97,0,2,3,128,0,11,3,249
,3,226,38,209,96,160,2,4,198,209,160,16
0,3
690 DATA 11,198,241,70,217,5,38,118,193,1
99,22,1,4,91,4,96,163,158,2,1,170,38,4,
192
700 DATA 2,2,226,38,216,0,140,2,16,0,216,
0,140,2,16,0,220,96,136,0,128,129,22,25
2
710 DATA 216,32,160,4,160,44,216,32,160,3
2,131,212,2,1,160,30,192,177,19,6,216,2
720 DATA 140,2,6,194,216,2,140,2,16,248,2
,0,0,91,2,1,208,0,216,0,140,2,6,192,216
,0
730 DATA 140,2,16,0,216,1,140,0,4,193,2,2
,3,0,2,0,0,88,216,0,140,2,6,192,216,0,1
40,2
740 DATA 6,193,216,1,140,0,6,193,5,129,12
8,129,22,249,2,0,0,64,2,1,226,38,2,2
750 DATA 250,38,216,0,140,2,6,192,216,0,1
40,2,16,0,216,49,140,0,128,129,22,252,2
,0
760 DATA 0,96,2,1,38,118,2,2,62,118,216,0
,140,2,6,192,216,0,140,2,16,0,216,49,14
0,0
770 DATA 128,129,22,252,216,32,160,26,131
,212,192,32,160,26,216,0,140,2,6,192,21
6,0
780 DATA 140,2,2,224,131,224,6,160,0,14,2
,224,112,184,152,32,131,117,160,200,19,
25
790 DATA 152,32,131,117,160,201,22,2,4,96
,164,76,152,32,131,117,160,202,22,2,4,9
6
800 DATA 164,72,152,32,131,117,160,203,22
,3,4,202,4,96,165,178,152,32,131,117
810 DATA 160,204,22,223,7,10,16,248,2,1,1
60,48,192,177,19,6,216,2,140,2,6,194,21
6,2
820 DATA 140,2,16,248,2,0,0,64,2,1,170,38
,2,2,226,38,216,0,140,2,6,192,216,0,140
,2
830 DATA 16,0,216,49,140,0,128,129,22,252
,4,192,216,0,131,124,4,91,211,160,160,1
3
840 DATA 9,142,2,142,0,32,27,2,4,96,163,1
58,211,224,160,12,19,6,2,13,9,0,2,142,0
,91
850 DATA 26,6,16,245,2,13,4,0,2,142,0,159
,27,240,2,46,255,224,10,62,163,78,211,1
60
860 DATA 160,15,9,142,2,142,0,32,27,230,6
,14,23,228,10,62,211,32,160,14,9,140,2,
140
870 DATA 0,24,27,221,6,12,23,219,195,12,1
9,4,2,46,1,0,6,12,22,252,210,96,160,17
880 DATA 4,202,210,160,160,16,11,202,226,
74,2,0,38,118,160,14,2,1,0,8,220,9,6,1
890 DATA 22,253,2,0,226,38,160,14,2,1,0,8
,6,205,216,13,140,2,6,205,216,13,140,2,
16,0
900 DATA 220,32,136,0,6,1,22,252,16,181,7
,10,16,1,4,202,216,32,152,2,160,24,216,
32
910 DATA 152,2,160,25,6,32,160,24,2,0,15,
128,2,1,160,84,2,2,0,40,4,32,96,40,2,9
920 DATA 15,137,200,9,131,86,4,32,96,56,0
,8,2,0,15,128,2,1,3,0,4,32,96,36,2,0,16
,0
930 DATA 2,1,160,212,2,2,0,8,4,32,96,40,2
,0,15,133,2,1,8,0,4,32,96,36,2,9,15,137
940 DATA 200,9,131,86,4,32,96,56,0,8,2,3,
0,32,2,4,227,30,193,68,2,0,16,0,2,1,160
,220
950 DATA 2,2,0,6,4,32,96,40,2,0,16,6,2,6,
0,192,2,7,0,8,208,85,194,138,19,1,5,65,
4,32
960 DATA 96,36,6,6,19,8,5,128,6,7,19,2,5,
133,16,243,2,37,0,249,16,238,2,0,15,133
,2,1
970 DATA 198,0,4,32,96,36,2,9,15,137,200,
9,131,86,4,32,96,56,0,8,6,3,19,3,2,36
980 DATA 255,248,16,207,2,0,16,0,2,1,160,
226,2,2,0,2,4,32,96,40,2,0,15,133,2,1,2
,0
990 DATA 4,32,96,36,200,9,131,86,4,32,96,
56,0,8,2,0,15,128,2,1,1,0,4,32,96,36,20
0,9
1000 DATA 131,86,4,32,96,56,0,8,2,0,15,12
8,2,1,241,166,2,2,2,0,4,32,96,40,216,32
1010 DATA 160,24,156,2,216,32,160,25,156,
2,4,96,163,36
1020 FOR I=-24550 TO -23177
1030 READ N
1040 CALL LOAD(I,N)
1050 NEXT I
```

Copie d'écran sur Seikosha GP250

Michel Schurmans

Nous vous proposons cette fois un sous-programme de copie d'écran pour imprimante Seikosha GP250 écrit en **Basic Etendu**. La présentation est celle d'une fonction à intégrer par *MERGE* dans le programme principal, ou à placer en fin de programme.



Les codes de la GP250 sont les suivants :

- **CHR$(27);"L";CHR$(x)**
 modifie l'interligne (x=2 : lignes jointives; x=3 : interligne normal);
- **CHR$(27);"G"**
 code introduisant le graphisme;
- **CHR$(0);CHR$(LEN(I$))**
 nombre de "colonnes" graphiques (soit nombre de caractères dans I$). Ce nombre étant, dans cette procédure, toujours inférieur à 255, le poids fort de ce nombre est toujours 0 (CHR$(0) est la seule valeur possible);
- **CHR$(15)**
 rétablit l'impression normale par suppression du mode graphique.

□

```
32709 ! **********************
32710 ! * Copie d'ecran sur   *
32711 ! *      imprimante     *
32712 ! *    Seikosha GP250   *
32713 ! **********************
32714 ! *     Basic Etendu    *
32715 ! **********************
32716 !
32717 ! CALL HARDCOPY(LigneDebut,LigneFin
,ColonneDebut,ColonneFin) : 1,24,1,32 p
our la totalite de l'ecran.
32718 !
32719 SUB HARDCOPY(LD,LF,CD,CF)
32720 !
32721 DIM CHA$(112),T(8,8)
32722 U$="123456789ABCDEF"
32723 FOR C=0 TO 112 :: CHA$(C)="" :: NEX
T C
32724 !
32725 OPEN #1:"PIO.LF.CR"
32726 PRINT #1:CHR$(27);"L";CHR$(2)! Lign
es jointives
32727 !
32728 FOR LC=LD TO LF
32729 I$=""
32730 FOR COL=CD TO CF
32731 !
32732 CALL GCHAR(LC,COL,CH)
32733 IF CH<32 THEN CH=32
32734 IF CHA$(CH-31)="" THEN 32736 ! Evit
e le recalcul des caracteres
32735 I$=I$&CHA$(CH-31):: GOTO 32760
32736 CALL CHARPAT(CH,A$)
32737 !
32738 ! Transformation du codage horizont
al du TI
32739 ! en une matrice T(,)
32740 !
32741 FOR L=1 TO 8
32742 N=POS(U$,SEG$(A$,2*L-1,1),1)*16+POS
(U$,SEG$(A$,2*L,1),1)
32743 V=128
32744 FOR C=1 TO 8
32745 IF N<V THEN T(L,C)=0 ELSE T(L,C)=1
:: N=N-V
32746 V=V/2
32747 NEXT C
32748 NEXT L
32749 !
32750 ! Code imprimante par codage vertic
al de la matrice
32751 !
32752 FOR C=1 TO 8
32753 N=0 :: V=1
32754 FOR L=1 TO 8 :: N=N+T(L,C)*V
32755 V=V*2
32756 NEXT L
32757 I$=I$&CHR$(N):: CHA$(CH-31)=CHA$(CH
-31)&CHR$(N)
32758 NEXT C
32759 !
32760 IF LEN(I$)<128 AND COL<CF THEN 3276
2
32761 PRINT #1:CHR$(27);"G";CHR$(0);CHR$(
LEN(I$));I$;:: I$=""
32762 NEXT COL
32763 PRINT #1:CHR$(15):"Texte eventuel"
32764 PRINT #1:CHR$(13);CHR$(10)
32765 NEXT LC
32766 PRINT #1:CHR$(15);CHR$(27);"L";CHR$
(3):: CLOSE #1
32767 SUBEND
```

La compilation séparée
Les Bibliothèques

Gérard Santraille

Comme nous avons tous pu le remarquer, la documentation fournie à chaque acquéreur d'une carte P-code présente de nombreux inconvénients. Citons les deux plus flagrants :

- elle est rédigée en anglais (notons au passage que la législation française est trés claire à ce sujet, et impose théoriquement que tout matériel vendu à l'intérieur de nos frontières soit accompagné d'une notice française...)
- elle est totalement incompréhensible par un utilisateur ne possédant pas une certaine expérience en la matière et ne permet pas, hélas, de s'initier au système-P et encore moins au Pascal.

Aussi, nous entamons ici une série d'articles destinée à familiariser le lecteur avec le système d'exploitation qui supporte le Pascal (ce fameux système-P).
Nous ne nous étendrons pas sur les différentes commandes permettant de créer et de compiler un programme (ce sont à peu près les seules notions qui soient clairement décrites dans le manuel), nous en avons d'ailleurs déjà parlé dans de précédents numéros.

Le système-P inclut de très puissantes caractéristiques que l'on retrouve dans l'exploitation de tout langage dit "évolué".

Avec un tel système d'exploitation, on peut créer et gérer des bibliothèques de programmes (contenant des modules écrits en assembleur ou en Pascal), on peut simuler un traitement multitâche où des procédures ne s'exécutent pas les unes après les autres (comme en Basic), mais conjointement; on peut encore modifier directement un code sans avoir à recompiler le programme. En un mot, on a accès à de très nombreuses ressources de la machine (pas à toutes : seul l'assembleur permet de TOUT faire).

L'apprentissage de toutes ces possibilités n'est pas aisé (où serait le plaisir) mais, au bout du compte, on est largement récompensé de l'investissement en temps..

Je vous propose aujourd'hui une petite promenade à travers deux des notions les plus complexes des langages de haut niveau. Il s'agit de la compilation séparée et des bibliothèques.

Avant d'aller plus avant, il convient d'assimiler la notion d'*UNIT*, qui est la base de cette technique de programmation. Le Pascal permet la compilation séparée et l'exécution d'un programme par morceaux. Prenons un exemple précis : le compilateur du *système-P*(SYSTEM.COMPILER) est lui même un programme écrit en Pascal dont la taille, sur la disquette, est de 99 blocs (soit prés de 50 Koctets). Il est donc impossible de le charger en entier dans la mémoire de l'ordinateur (notons au passage qu'une fois le système d'exploitation chargé, il reste bien moins de 48Ko libres pour un programme utilisateur, une grosse trentaine tout au plus).

Le compilateur est donc découpé en blocs logiques (appelé *UNITs*) qui sont chargés, exécutés puis ôtés de la mémoire au fur et à mesure que l'ordinateur en a besoin. Par cette technique (appelée *swapping* ou *overlay*) on peut faire tourner de très gros programmes avec une petite Unité Centrale (UC).
La grande majorité des systèmes d'exploitation utilisent cette technique.
La compilation séparée apporte également une plus grande souplesse lors du développement d'un programme. Un programme de 1000 lignes que l'on découpe en dix *UNITs* de 100 lignes est d'une maintenance bien plus aisée.

Lors d'une modification, seul le "morceau" concerné devra être recompilé (c'est appréciable quand on connaît la lenteur du compilateur du TI). Le déverminage (recherche des erreurs, ou *debugging*) d'un programme s'en trouve également grandement facilité.

Avec le système-P, la compilation séparée se construit donc avec des entités appelées *UNITs* qui sont elles-mêmes des ensembles de segments. Différentes *UNITs* sont alors rassemblées dans un même fichier code. Elles constituent alors une librairie. Ainsi les différentes routines permettant de gérer les sons, d'animer les lutins ou de redéfinir les caractères, sont regroupées dans différentes *UNITs* incluses dans la librairie *SYSTEM.LIBRARY*. L'intérêt d'une telle structure apparaît dès que l'on désire utiliser souvent une procédure.

Supposons que vous soyez un mathématicien accompli, et que vous ayez écrit un ensemble de routines calculant la plupart des fonctions classiques de trigonométrie hyperbolique. Pour pouvoir utiliser ces fonctions dans n'importe quel programme, il suffit d'inclure dans le source du programme :

```
(*$u nom-de-la-librairie *)
uses nom-de-l'unit ;
```

Par ces simples déclarations, tous vos programmes auront accès aux différentes fonctions. On est bien loin du Basic qui ne possède aucune structure comparable.

Après ces considérations générales, voyons des exemples concrets.

L'autre notion fondamentale de gestion de la mémoire est le segment. Il s'agit d'une entité logique (comme une procédure ou une fonction) dont le code ne sera chargé en mémoire qu'au moment de son exécution. Celle-ci terminée, la mémoire occupée sera restituée au système.

La déclaration d'un segment se fait très simplement au moyen du mot-clé *SEGMENT* selon :

```
SEGMENT PROCEDURE TOTO;

BEGIN
END;
```

Une première remarque s'impose; il convient de ne pas confondre la notion de *segment* avec celle d'*unit*.
Le source d'un segment doit se trouver **avec** celui du programme principal. Un segment (contrairement à une unit) est compilé en même temps que le reste du programme.

Par contre, il n'est pas chargé en même temps que le programme. Pour s'en convaincre, écrivons le petit programme suivant :

```
PROGRAM TEST;

SEGMENT PROCEDURE TOTO;
BEGIN
WRITELN('TOTO')
END;

BEGIN
WRITELN('PROGRAMME PRINCIPAL')
READLN;
TOTO
END.
```

Lors de l'exécution du programme, l'ordinateur ne va pas charger la procédure *TOTO*. Il écrit PROGRAMME *PRINCIPAL* et attend que vous appuyiez sur *ENTER* (READLN;). Si l'on presse sur *ENTER*, le lecteur de disquette se met effectivement en marche et charge le segment *TOTO*.

Remarque technique : les segments doivent être déclarés avant les procédures classiques. Si une procédure-segment appelle une procédure non-segment cette dernière doit être déclarée *FORWARD* selon :

```
PROGRAM TEST2;
PROCEDURE PR2; FORWARD;
SEGMENT PROCEDURE PR1;
  BEGIN (* DEBUT DE PR1*)
  ...
  ...
  END;
    PROCEDURE PR2;
    BEGIN (* DEBUT DE PR2 *)
    ...
    ...
    END;
  SEGMENT PROCEDURE PR3;
  BEGIN (* DEBUT DE PR3 *)
  ...
  ...
  END;
BEGIN (* DEBUT DE TEST2 *)
END.
```

Détaillons maintenant la structure des units. Bien que beaucoup plus complexe, elle se décompose en cinq parties distinctes : l'en-tête, l'interface, l'implémentation, l'initialisation et la terminaison.

L'en-tête est constitué par le mot-clé *UNIT* suivi de son nom.

L'interface contient les déclarations (procédures ou fonctions) qui pourront être utilisées par un programme faisant référence à cette unit. Cette section est souvent appelée partie publique de l'unit. Elle se compose du mot-clé *INTERFACE* (sans ";") et du nom de toutes les procédures.

L'implémentation est constituée par tous les corps de toutes les procédures ou fonctions déclarées dans l'interface. Elle débute par le mot-clé *IMPLEMENTATION*.

Les deux dernières parties (*INITIALISATION* et *TERMINAISON*) contiennent un bloc d'instructions Pascal entre un *BEGIN* et un *END*. Ces instructions permettent d'initialiser et de terminer (si nécessaire) les diverses composantes de l'unit. La partie *INITIALISATION* est séparée de la partie *TERMINAISON* par la ligne "***;"

Ces deux sections peuvent être vides toutes les deux. Remarque importante : ces deux sections d'instructions s'exécutent avant (respectivement après) toute instruction du programme faisant référence à l'unit, et non pas à chaque appel d'un de ses éléments.

Un grand classique du Pascal (la fonction factorielle) dans une unit s'écrirait de la façon suivante:

```
UNIT MATH;
INTERFACE
FUNCTION FACTORIELLE
(I:INTEGER):INTEGER;
IMPLEMENTATION
FUNCTION FACTORIELLE;
BEGIN
IF N=0 THEN FACTORIELLE:=1
ELSE FACTORIELLE:=FACTORIELLE(N-1)
END;
BEGIN
***;
END.
```

Après compilation, le fichier code généré sera considéré comme une bibliothèque et pourra donc être utilisé en temps que tel. On pourra alors utiliser la fonction factorielle en déclarant dans le programme appelant :

```
(*$u nom-du-fichier-code)
uses math;
```

N'oubliez pas que, lors de la compilation, la bibliothèque doit être accessible par le compilateur.

Lors de l'exécution, l'ordinateur doit savoir dans quelle bibliothèque se trouve l'unit math : la directive (*$u nom-du-fichier-code *) ne lui suffit pas.

Le nom de ce fichier doit se trouver dans un fichier texte qui a par defaut le nom **user.lib* (l'étoile représente le volume racine, c'est-à-dire celui qui a démarré le système. On peut changer le nom de ce fichier code en faisant en mode commande : X pour eXecute puis en tapant L=nouveau-nom, nouveau-nom se substituera alors à *USERLIB.TEXT*.

Tout ceci peut paraître bien lourd mais, avec un peu de pratique, on manie très rapidement les bibliothèques. Une fois ces mécanismes acquis, on ne conçoit plus la programmation en Pascal sans l'aide des bibliothèques.

Résumons les différentes étapes nécessaires à l'emploi d'une bibliothèque :

- créer, avec l'éditeur, l'unit;
- la compiler;
- mettre le nom de la bibliothèque (il s'agit du nom du fichier) dans un fichier text s'appelant *userlib.text* et situé sur le volume racine;
- déclarer à nouveau le nom de la bibliothèque dans le programme utilisant l'unit par la directive (*$U NOM-DE-LA-BIBLI*);
- déclarer l'utilisation de l'unit par *USES NOM-DE-L'UNIT*.

Pour utiliser plus simplement une unit, il existe une autre technique : on peut la mettre dans la librairie standard du système *SYSTEM.LIBRARY qui n'a pas à être déclarée (il sera quand même nécessaire de mettre dans le programme appelant USES NOM-DE-L'UNIT;).

C'est l'utilitaire *LIBRARY* qui se charge de cette tâche. Ce programme permet par ailleurs de grouper en une

seule librairie plusieurs units.

Son utilisation est assez délicate et fera certainement l'objet d'un article détaillé dans un prochain numéro. Supposons que le fichier sous-lequel nous avons sauvegardé notre unit math s'appelle *mathunit* (quelle imagination !). Lançons le programme *library* (il se trouve sur la disquette contenant tous les utilitaires) avec eXecute en mode commande.
La première question est *Output file ?* Pour ajouter une unit à la bibliothèque système tapons *LIBRARY2. Nous changerons plus tard le nom de ce fichier en *SYSTEM.LIBRARY. S'il s'agit de la création d'une nouvelle bibliothèque il faut taper le nom de celle-ci.
Le programme ouvre alors le fichier concerné comme fichier de sortie : il y mettra toutes les units que nous voudrons. Dans un premier temps nous allons mettre dans *LIBRARY2 toutes celles de *SYSTEM.LIBRARY. Nous spécifions donc *SYSTEM.LIBRARY comme *input file ?*, qui est la question suivante. Le programme affiche alors la liste de son contenu (7 units). Nous pressons ensuite la lettre *E* pour E(very, et tous les units de *SYSTEM.LIBRARY (input file) sont copiés sur *LIBRARY2 (output file). Une fois le transfert accompli, nous sélectionnons un autre fichier d'entrée en pressant *N* pour New, nous sélectionnons bien entendu *MATHUNIT puis *E* à nouveau pour copier toutes les units de *MATHUNIT* (dans notre cas, il n'y en a qu'une) sur *LIBRARY2*. Le nouveau contenu de cette dernière s'affiche alors et on s'aperçoit qu'elle contient bien *MATH*. On a donc bien créé une bibliothèque unique regroupant les units de *SYSTEM. LIBRARY et de *MATHUNIT.

Il ne nous reste plus qu'à détruire les deux anciennes et à changer (avec le Filer) le nom de *LIBRARY2 en *SYSTEM. LIBRARY.

Nous voilà au terme de notre première excursion dans le monde étrange de la compilation séparée. Ne vous inquiétez pas si certains points vous semblent encore obscurs. Laissez décanter et persévérez...

□

Trouvez votre chemin dans les jeux d'aventure TI

Denise Amrouche

Vous êtes l'heureux possesseur du module "Adventure" et d'une ou plusieurs cassettes de jeux d'aventure, mais le vocabulaire anglais vous arrête un peu. Nous vous proposons deux programmes pour la mini-mémoire, qui vous permettrons d'afficher sur l'écran ou d'écrire sur votre imprimante la partie texte contenue dans les cassettes et, en particulier, le vocabulaire connu par le jeu. Tout ceci devrait vous permettre d'explorer ensuite les aventures plus facilement.

Affichage sur l'écran

Mode d'emploi

- insérer la Mini-mémoire et charger, avec "EASY BUG" ou l'assembleur ligne par ligne, le programme suivant;
- charger la cassette de jeu voulue en Basic (OLD CS1) sans se préoccuper du message d'erreur final;
- taper CALL LINK("LAVENT") puis faire "ENTER";
- le contenu texte de la cassette s'affiche alors écran par écran;
- vous passez d'un écran au suivant en appuyant sur une touche quelconque;
- arrivé au dernier écran, vous pouvez revoir le contenu de la même cassette en recommençant au point 3.

Plan du texte

- introduction commune à tous les jeux d'aventure;
- partie incompréhensible qui correspond probablement à un programme en langage machine;
- long texte regroupant les morceaux de phrases que vous voyez apparaître sur l'écran en majuscules, par les trois ou quatre premières lettres de chaque nom connu par le jeu;
- une autre partie de texte en majuscules, correspondant aux trois ou quatre premières lettres des verbes connus par le jeu.

Schéma du programme

Le contenu de la cassette de jeu se charge en mémoire vidéo à partir de l'adresse >0709, jusqu'à l'adresse >2E3F au maximum semble-t-il, pour les jeux que j'ai pu tester. Il est lu par tranches de 28 octets et chaque octet est examiné. S'il correspond à un code ASCII affichable, le caractère correspondant est mis dans la table d'écran, alors qu'il est remplacé par un espace dans le cas contraire. On remplit ensuite la page écran complète et on attend qu'une touche soit enfoncée pour recommencer avec la page suivante.

Voir listing 1

Ecriture sur l'imprimante

Mode d'emploi

- insérer la Mini-mémoire et charger le programme suivant;
- charger la cassette de jeu voulue en Basic (OLD CS1) sans se préoccuper du message d'erreur final;
- taper CALL LINK("IAVENT") puis "ENTER";
- le texte s'imprime.

Le programme est écrit pour une imprimante EPSON branchée en parallèle. □

Listing 2 (imprimante)

```
7D00                AORG >7E00

          *TAMP. EN >7DFE SUR 80 OCTETS

          *L'ADRESSE >7DA0 EST UTILISEE

7E00 D820           MOVB @>9802,@>7DFE
7E02 9802
7E04 7DFE
7E06 06E0           SWPB @>7DFE
7E08 7DFE
7E0A D820           MOVB @>9802,@>7DFE
7E0C 9802
7E0E 7DFE
7E10 06E0           SWPB @>7DFE
7E12 7DFE
7E14 0620           DEC  @>7DFE
7E16 7DFE

7E18 0200           LI   R0,>3000
7E1A 3000
7E1C 0201           LI   R1,PD
7E1E 7EF6
7E20 0202           LI   R2,13
7E22 000D
7E24 0420           BLWP @>6028  (VMBW)
7E26 6028
7E28 06A0           BL   @IM
7E2A 7EDA

7E2C 0200           LI   R0,>3000
7E2E 3000
7E30 0201           LI   R1,>0300
7E32 0300
7E34 0420           BLWP @>6024  (VSBW)
7E36 6024

7E38 0200           LI   R0,>3005
7E3A 3005
7E3C 0201           LI   R1,>5000
7E3E 5000
7E40 0420           BLWP @>6024  (VSBW)
7E42 6024

7E44 0204           LI   R4,>6B8
7E46 06B8

7E48 0224  B0       AI   R4,>50
7E4A 0050
7E4C C004           MOV  R4,R0
7E4E 0201           LI   R1,>7DA0
7E50 7DA0
7E52 0202           LI   R2,80
7E54 0050
7E56 0420           BLWP @>6030  (VMBR)
7E58 6030

7E5A C151  B1       MOV  *R1,R5
7E5C 04C7           CLR  R7
7E5E 9805           CB   R5,@CO
7E60 7EF2
7E62 1235           JLE  BL
7E64 9805           CB   R5,@CO+1
7E66 7EF3
7E68 1B32           JH   BL

7E6A 06C5  B2       SWPB R5
7E6C 0587           INC  R7
7E6E 9805           CB   R5,@CO
7E70 7EF2
7E72 122D           JLE  BL
7E74 9805           CB   R5,@CO+1
7E76 7EF3
7E78 1B2A           JH   BL

7E7A 06C5  B3       SWPB R5
7E7C CC45           MOV  R5,*R1+
7E7E 0281           CI   R1,>7DFE
7E80 7DFE
7E82 11EB           JLT  B1

7E84 0200           LI   R0,>3100
7E86 3100
7E88 0201           LI   R1,>7DA0
7E8A 7DA0
7E8C 0202           LI   R2,80
7E8E 0050
7E90 0420           BLWP @>6028  (VMBW)
7E92 6028
7E94 06A0           BL   @IM
7E96 7EDA
7E98 0284           CI   R4,>2E3F
7E9A 2E3F
7E9C 1501           JGT  FI
7E9E 10D4           JMP  B0

7EA0 0200  FI       LI   R0,>1D00
7EA2 1D00
7EA4 0201           LI   R1,>0100
7EA6 0100
7EA8 0420           BLWP @>6024  (VSBW)
7EAA 6024
7EAC 06A0           BL   @IM
7EAE 7EDA
7EB0 D820           MOVB @>7DFE,@>9C02
7EB2 7DFE
7EB4 9C02
7EB6 06E0           SWPB @>7DFE
7EB8 7DFE
7EBA D820           MOVB @>7DFE,@>9C02
7EBC 7DFE
7EBE 9C02
7EC0 04C0           CLR  R0
7EC2 D800           MOVB R0,@>837C
7EC4 837C
7EC6 02E0           LWPI >83E0
7EC8 83E0
7ECA 0460           B    @>0070
7ECC 0070

7ECE D160  BL       MOVB @CO+2,R5
7ED0 7EF4
7ED2 0287           CI   R7,0
7ED4 0000
7ED6 13C9           JEQ  B2
7ED8 10D0           JMP  B3

7EDA 020A  IM       LI   R10,>3009
7EDC 3009
7EDE C80A           MOV  R10,@>8356
7EE0 8356
7EE2 0420           BLWP @>6038  (DSRLNK)
7EE4 6038
7EE6 0008           DATA 8
7EE8 020A           LI   R10,50
7EEA 0032
7EEC 060A           DEC  R10
7EEE 16FE           JNE  $-2
7EF0 045B           B    *R11
7EF2 207E  CO       DATA >207E,>2000
```

```
7EF4 2000
7EF6 0002   PD       DATA >0002,>3100,>5000,
7EF8 3100                 >0000,>0003
7EFA 5000
7EFC 0000
7EFE 0003
7F00 5049            TEXT 'PIO.'
7F02 4F2E

7F04                 AORG >7FF8

7FF8 4941            TEXT 'IAVENT'
7FFA 5645
7FFC 4E54
7FFE 7E00            DATA >7E00

8000                 AORG >701C

701C 7F04            DATA >7F04,>7FF8
701E 7FF8            .
                     END
```

Listing 1 (écran)

```
                     AORG >7E00
7E00 0204            LI   R4,>6ED
7E02 06ED
7E04 0206            LI   R6,2
7E06 0002
7E08 207E   CO       DATA >207E
7E0A 8000            DATA >8000
7E0C 04C0            CLR  R0
7E0E 0201            LI   R1,>8000
7E10 8000
7E12 0202            LI   R2,768
7E14 0300
7E16 0420            BLWP @>6024
7E18 6024
7E1A 1002            JMP  S1
7E1C D801   S2       MOVB R1,@>8C00
7E1E 8C00
7E20 0602   S1       DEC  R2
7E22 16FC            JNE  S2
7E24 0224   B0       AI   R4,>1C
7E26 001C
7E28 C004            MOV  R4,R0
7E2A 0201            LI   R1,>7DD8
7E2C 7DD8
7E2E 0202            LI   R2,28
7E30 001C
7E32 0420            BLWP @>6030
7E34 6030
7E36 C151   B1       MOV  *R1,R5
7E38 04C7            CLR  R7
7E3A 9805            CB   R5,@CO
7E3C 7E08
7E3E 122B            JLE  BL
7E40 9805            CB   R5,@CO+1
7E42 7E09
7E44 1B28            JH   BL
7E46 0225            AI   R5,>6000
7E48 6000
7E4A 06C5   B2       SWPB R5
7E4C 0587            INC  R7
7E4E 9805            CB   R5,@CO
7E50 7E08
7E52 1221            JLE  BL
7E54 9805            CB   R5,@CO+1
7E56 7E09
7E58 1B1E            JH   BL
7E5A 0225            AI   R5,>6000
7E5C 6000
7E5E 06C5   B3       SWPB R5
7E60 CC45            MOV  R5,*R1+
7E62 0281            CI   R1,>7E00
7E64 7E00
7E66 11E7            JLT  B1
7E68 C006            MOV  R6,R0
7E6A 0201            LI   R1,>7DD8
7E6C 7DD8
7E6E 0202            LI   R2,28
7E70 001C
7E72 0420            BLWP @>6028
7E74 6028
7E76 0284            CI   R4,>2E3F
7E78 2E3F
7E7A 1506            JGT  FI
7E7C 0226            AI   R6,32
7E7E 0020
7E80 0286            CI   R6,>2FF
7E82 02FF
7E84 150E            JGT  SC
7E86 10CE            JMP  B0
7E88 04C0   FI       CLR  R0
7E8A D800            MOVB R0,@>837C
7E8C 837C
7E8E 02E0            LWPI >83E0
7E90 83E0
7E92 0460            B    @>0070
7E94 0070
7E96 D160   BL       MOVB @CO+2,R5
7E98 7E0A
7E9A 0287            CI   R7,0
7E9C 0000
7E9E 13D5            JEQ  B2
7EA0 10DE            JMP  B3
7EA2 0206   SC       LI   R6,2
7EA4 0002
7EA6 04E0            CLR  @>8374
7EA8 8374
7EAA 0420            BLWP @>6020
7EAC 6020
7EAE D360            MOVB @>837C,R13
7EB0 837C
7EB2 024D            ANDI R13,>FF00
7EB4 FF00
7EB6 020E            LI   R14,>2000
7EB8 2000
7EBA 234E            COC  R14,R13
7EBC 16F2            JNE  SC
7EBE D020            MOVB @>8375,R0
7EC0 8375
7EC2 0240            ANDI R0,>FF00
7EC4 FF00
7EC6 0280            CI   R0,>FF00
7EC8 FF00
7ECA 13EB            JEQ  SC
7ECC 10AB            JMP  B0

                     AORG >7FF8

7FF8 4C41            TEXT 'LAVENT'
7FFA 5645
7FFC 4E54
7FFE 7E00            DATA >7E00

                     AORG >701C

701C 7ECE            DATA >7ECE,>7FF8
701E 7FF8
```

Srand

Georges Goument

NDLR: ce programme peut être sauvegardé sur disquette (avec CALL FILES (1)), mais ne fonctionne pas si le contrôleur est raccordé.

Ce jeu, écrit en **Basic TI**, consiste à éliminer tous les pions de son adversaire. Chaque joueur déplace alternativement ses pions en suivant colonnes ou diagonales, mais jamais une ligne, en jeu normal.

Le plateau de jeu est formé de 9 lignes, 9 colonnes et 14 diagonales. Dans la pratique, les lignes et colonnes 2, 4, 6 et 8 n'apparaissent pas.

Les pions ne peuvent reculer ou se déplacer latéralement que dans le cas d'une prise de pion adverse. Chaque pion pris est retiré du jeu. On est obligé de prendre, et toujours le maximum de pions.

La *rafle*, ou prise multiple, est annoncée par le joueur qui l'effectue. Il doit dire le nombre de pions qu'il compte prendre et, si l'adversaire lui démontre qu'il pouvait en prendre plus, il souffle le pion devant effectuer la rafle et prend le tour de jeu de son adversaire. Contrairement au jeu de dames, lors d'une rafle, les pions sont pris un par un et retirés du jeu au fur et à mesure.

Lorsqu'un pion atteint la ligne adverse la plus reculée, ce pion est promu *Sultan*. Le sultan se déplace en toutes directions, sur n'importe quelle distance, et prend les pions adverses placés sur sa trajectoire à condition que la case située immédiatement après soit libre.

Chaque partie débute par une rencontre qui est conventionnellement toujours identique (voir détails plus loin). Ce sont les pions jaunes qui débutent cette rencontre.

Le Srand avec le TI

La couleur du pion qui joue et les différentes phases d'un coup joué sont affichées à gauche pour les pions noirs, à droite pour les pions jaunes. Un symbole pour chacune des phases du jeu, ainsi que le clignotement du pion joué, et du pion pris s'il y a une prise, offrent une meilleure vue d'ensemble dans le déroulement d'un coup.

Les symboles sont les suivants :

- **D** noir sur fond rouge - pion à déplacer;
- **P** noir sur fond cyan - pion à prendre (en cas de prise);
- **A** noir sur fond vert - case d'arrivée.

Ces symboles s'affichent également en regard des numéros de ligne et de colonne choisis (coordonnées des pions joués).
Important : tapez toujours le numéro de ligne en premier.

Déplacement d'un pion sans prise

Taper les deux coordonnées D, appuyer sur la touche "N", taper les deux coordonnées A. Le pion effectue son déplacement.

Prise d'un pion

Taper les deux coordonnées D, les deux coordonnées du pion à prendre puis celles de la case A. Le pion quitte sa case, prend le pion adverse et occupe sa nouvelle place.

Rafle ou prise multiple

Taper les coordonnées D, appuyer sur "R", puis taper le nombre de pions à prendre (touches 2 à 9). Si votre adversaire est d'accord, taper les deux coordonnées P, puis A. Lorsque chaque pion est pris, recommencer D, P et A autant de fois que nécessaire; ne plus s'occuper de la touche "R".

Pion soufflé

Une pression sur la touche "S" provoque le changement de joueur. Taper ensuite uniquement D et P. Le joueur qui souffle garde la main et rejoue normalement.

Annulation

La touche "A" permet de jouer un autre pion. Cette annulation est faite automatiquement en cas d'erreur.

Fin de partie

La touche "I" arrête la partie en cours.

Récapitulatif des touches

N : pas de prise;
R : rafle; minimum 2, maximum 9 (si plus, recommencer);
S : pion soufflé;
A : annulation d'un coup;
I : fin de partie;
1 à 9 : coordonnées X et Y des pions;
2 à 9 avec R : nombre de pions raflés.

Le jeu avec un Sultan se pratique de la même façon, seuls les déplacements ne sont plus limités.

Déroulement d'une rencontre en 12 coups

Taper les 2 chiffres sous chaque symboles; cette rencontre permet de se familiariser avec le jeu et ses différents modes d'affichages.

Symboles ->		D		P	A
1	Jaune	4 4		"N"	5 5
2	Noir	6 6		5 5	4 4
3	Jaune	3 3		4 4	5 5
4	Noir	5 1		4 2	3 3
5	Jaune	2 2		3 3	4 4
6	Noir	5 3		4 3	3 3
7	Jaune	5 5	R 2	5 4	5 3
8	Jaune	5 3		5 2	5 1
9	Noir	3 3		4 4	5 5
10	Jaune	5 6		5 5	5 4
11	Noir	6 4		5 4	4 4
12	Jaune	4 5		4 4	4 3

Fin de la rencontre; les pions noirs commencent la partie libre.

□

```
100 REM ****************
110 REM *              *
120 REM *     SRAND    *
130 REM *              *
140 REM ****************
150 REM *              *
160 REM *   BASIC  TI  *
170 REM *              *
180 REM ****************
190 REM *              *
200 REM *   COPYRIGHT  *
210 REM *              *
220 REM *  99 MAGAZINE *
230 REM *              *
240 REM *       &      *
250 REM *              *
260 REM *    GEORGES   *
270 REM *              *
280 REM *    GOUMENT   *
290 REM *              *
300 REM ****************
310 GOSUB 7000
320 PRINT TAB(11);"LE SRAND"::TAB(11);"@@
@@@@@@":::
330 PRINT " Ce jeu national Mauritanienes
t une variante elaboree dujeu de dames.
"::
340 PRINT "           Les possibilites deco
mbinaisons sont multiples ce qui le ren
d fascinant et"
350 PRINT "vous garantis un agreable    pa
sse-temps.":::
360 FOR D=1 TO 800
370 NEXT D
380 CALL CLEAR
390 FOR I=33 TO 39
400 READ A$
410 CALL CHAR(I,A$)
420 NEXT I
430 DATA "FFFFFFFFFFFFFFFF","0000001F1010
101","000000FF","000000F01010101"
440 DATA "1010101010101010","1010101F","1
01010F"
450 FOR I=88 TO 105
460 READ A$
470 CALL CHAR(I,A$)
480 NEXT I
490 DATA "0000001F10141211","000000FF","0
00000FF00000001","000000FF10549211","10
1010101010101"
500 DATA "8040201008040201","000000000000
0001","0204081020408","8244281028448201
","1010101010101011"
510 DATA "1214101F10141211","925410FF0054
9211","1214101F","925410FF"
520 DATA "000000F01050901","905010F","905
010F01050901","0"
530 FOR I=40 TO 47
540 READ A$
550 CALL CHAR(I,A$)
560 NEXT I
570 DATA "005A81183C66FF","00FF663C18815A
","4462705A5A706244","22460E5A5A0E4622"
580 DATA "80423134041C4221","01428C2C2038
4284","844238202C8C4201","21421C0434314
28"
590 CALL CHAR(64,"7C24243C30282424")
600 CALL CHAR(66,"3C24203C0404243C")
610 CALL CHAR(67,"3C2020382020203C")
620 CALL CHAR(68,"2232322A2A262622")
630 CALL CHAR(128,"007C242424247C")
640 CALL CHAR(136,"007C24243C202")
650 CALL CHAR(144,"003C24243C2424")
660 CALL CHAR(120,"3C7EFFFFFFFF7E3C")
670 CALL CHAR(112,"3C7EFFFFFFFF7E3C")
680 CALL CHAR(152,"FFC3DFC3FBFBC3FF")
690 CALL CHAR(153,"003C203C04043C")
700 PRINT TAB(6);"1 2 3 4 5 6 7 8 9"::TAB
(4);"9";TAB(24);"9"::TAB(4);"8";TAB(24)
;"8"::TAB(4);"7";TAB(24);"7"::
710 PRINT TAB(4);"6";TAB(24);"6"::TAB(4);
"5";TAB(24);"5"::TAB(4);"4";TAB(24);"4"
::TAB(4);"3";TAB(24);"3"::
720 PRINT TAB(4);"2";TAB(24);"2"::TAB(4);
"1";TAB(24);"1"::TAB(6);"1 2 3 4 5 6 7
8 9":::
730 FOR I=8 TO 24
740 CALL VCHAR(3,I,105,17)
750 NEXT I
760 FOR I=7 TO 26 STEP 2
770 CALL HCHAR(1,I,33)
780 CALL HCHAR(21,I,33)
790 NEXT I
800 FOR I=2 TO 20 STEP 2
810 CALL VCHAR(I,6,33)
820 CALL VCHAR(I,26,33)
830 NEXT I
840 CALL HCHAR(1,6,33)
850 CALL HCHAR(1,26,33)
860 CALL HCHAR(21,6,33)
870 CALL HCHAR(21,26,33)
880 CALL HCHAR(2,7,44)
890 CALL HCHAR(2,8,40,17)
900 CALL HCHAR(2,25,45)
910 CALL VCHAR(3,25,42,17)
920 CALL HCHAR(20,25,46)
930 CALL HCHAR(20,8,41,17)
940 CALL HCHAR(20,7,47)
950 CALL VCHAR(3,7,43,17)
960 FOR I=8 TO 24 STEP 2
970 FOR Y=3 TO 11 STEP 2
980 CALL HCHAR(Y,I,112)
990 NEXT Y
1000 FOR Y=13 TO 19 STEP 2
1010 CALL HCHAR(Y,I,120)
1020 NEXT Y
1030 NEXT I
1040 FOR I=18 TO 24 STEP 2
1050 CALL HCHAR(11,I,120)
1060 NEXT I
1070 FOR I=9 TO 24 STEP 4
1080 FOR Y=3 TO 19 STEP 4
1090 CALL HCHAR(Y,I,89)
1100 NEXT Y
1110 FOR Y=4 TO 16 STEP 4
1120 CALL HCHAR(Y,I,93)
1130 NEXT Y
1140 FOR Y=5 TO 17 STEP 4
1150 CALL HCHAR(Y,I,94)
1160 NEXT Y
1170 FOR Y=6 TO 18 STEP 4
1180 CALL HCHAR(Y,I,95)
1190 NEXT Y
1200 NEXT I
1210 FOR I=10 TO 22 STEP 4
1220 FOR Y=4 TO 16 STEP 4
1230 CALL HCHAR(Y,I,94)
1240 NEXT Y
1250 NEXT I
1260 FOR I=11 TO 23 STEP 4
1270 CALL HCHAR(19,I,89)
1280 FOR Y=3 TO 15 STEP 4
1290 CALL HCHAR(Y,I,90)
1300 NEXT Y
1310 FOR Y=4 TO 16 STEP 4
```

```
1320 CALL HCHAR(Y,I,95)
1330 NEXT Y
1340 FOR Y=6 TO 18 STEP 4
1350 CALL HCHAR(Y,I,93)
1360 NEXT Y
1370 NEXT I
1380 FOR I=8 TO 20 STEP 4
1390 FOR Y=6 TO 18 STEP 4
1400 CALL HCHAR(Y,I,97)
1410 NEXT Y
1420 NEXT I
1430 FOR I=8 TO 24 STEP 4
1440 FOR Y=4 TO 16 STEP 4
1450 CALL HCHAR(Y,I,92)
1460 NEXT Y
1470 NEXT I
1480 FOR I=4 TO 18 STEP 2
1490 CALL VCHAR(I,24,92)
1500 NEXT I
1510 CALL HCHAR(11,16,96)
1520 CALL COLOR(1,10,1)
1530 CALL COLOR(2,16,10)
1540 CALL COLOR(3,5,10)
1550 CALL COLOR(4,5,10)
1560 CALL COLOR(5,4,1)
1570 CALL COLOR(13,2,9)
1580 CALL COLOR(14,2,8)
1590 CALL COLOR(15,2,3)
1600 CALL COLOR(16,11,2)
1610 CALL COLOR(11,2,8)
1620 CALL COLOR(12,11,8)
1630 FOR I=8 TO 10
1640 CALL COLOR(I,13,8)
1650 NEXT I
1660 CALL HCHAR(22,15,34)
1670 CALL HCHAR(22,16,35)
1680 CALL HCHAR(22,17,36)
1690 CALL HCHAR(23,15,37)
1700 CALL HCHAR(23,17,37)
1710 CALL HCHAR(24,15,38)
1720 CALL HCHAR(24,16,35)
1730 CALL HCHAR(24,17,39)
1740 XD=0
1750 XP=0
1760 U1=48
1770 U2=52
1780 V1=48
1790 V2=52
1800 W=0
1810 N=0
1820 R=0
1830 CS=0
1840 IF W=1 THEN 1900
1850 CJ=28
1860 Q=42
1870 O=25
1880 Z=120
1890 GOTO 1940
1900 CJ=4
1910 Q=43
1920 O=7
1930 Z=112
1940 CALL SOUND(10,660,5)
1950 CALL SOUND(10,550,5)
1960 CALL HCHAR(13,CJ,Z)
1970 IF N=2 THEN 2020
1980 IF N=4 THEN 2050
1990 P=128
2000 CC=15
2010 GOTO 2070
2020 P=136
2030 CC=17
2040 GOTO 2070
2050 P=144
2060 CC=19
2070 CALL HCHAR(CC,CJ,P)
2080 CALL KEY(3,T,S)
2090 IF R<>1 THEN 2110
2100 CALL HCHAR(23,16,PP)
2110 IF XD=0 THEN 2350
2120 CALL HCHAR(YD,XD,105)
2130 IF W=1 THEN 2190
2140 IF SJ=1 THEN 2170
2150 CALL HCHAR(YD,XD,120)
2160 GOTO 2230
2170 CALL HCHAR(YD,XD,152)
2180 GOTO 2230
2190 IF SJ=1 THEN 2220
2200 CALL HCHAR(YD,XD,112)
2210 GOTO 2230
2220 CALL HCHAR(YD,XD,153)
2230 IF XP=0 THEN 2350
2240 CALL HCHAR(YP,XP,105)
2250 IF W=1 THEN 2310
2260 IF SP=1 THEN 2290
2270 CALL HCHAR(YP,XP,112)
2280 GOTO 2350
2290 CALL HCHAR(YP,XP,153)
2300 GOTO 2350
2310 IF SP=1 THEN 2340
2320 CALL HCHAR(YP,XP,120)
2330 GOTO 2350
2340 CALL HCHAR(YP,XP,152)
2350 IF S=0 THEN 2080
2360 CALL HCHAR(23,16,32)
2370 IF T<>49 THEN 2430
2380 IF CS=1 THEN 2410
2390 Y=19
2400 GOTO 3750
2410 X=8
2420 GOTO 3810
2430 IF T<>50 THEN 2490
2440 IF CS=1 THEN 2470
2450 Y=17
2460 GOTO 3750
2470 X=10
2480 GOTO 3810
2490 IF T<>51 THEN 2550
2500 IF CS=1 THEN 2530
2510 Y=15
2520 GOTO 3750
2530 X=12
2540 GOTO 3810
2550 IF T<>52 THEN 2610
2560 IF CS=1 THEN 2590
2570 Y=13
2580 GOTO 3750
2590 X=14
2600 GOTO 3810
2610 IF T<>53 THEN 2670
2620 IF CS=1 THEN 2650
2630 Y=11
2640 GOTO 3750
2650 X=16
2660 GOTO 3810
2670 IF T<>54 THEN 2730
2680 IF CS=1 THEN 2710
2690 Y=9
2700 GOTO 3750
2710 X=18
2720 GOTO 3810
2730 IF T<>55 THEN 2790
2740 IF CS=1 THEN 2770
2750 Y=7
```

```
2760 GOTO 3750
2770 X=20
2780 GOTO 3810
2790 IF T<>56 THEN 2850
2800 IF CS=1 THEN 2830
2810 Y=5
2820 GOTO 3750
2830 X=22
2840 GOTO 3810
2850 IF T<>57 THEN 2910
2860 IF CS=1 THEN 2890
2870 Y=3
2880 GOTO 3750
2890 X=24
2900 GOTO 3810
2910 IF T<>82 THEN 3330
2920 CALL HCHAR(23,16,64)
2930 R=1
2940 CALL SOUND(50,550,10)
2950 CALL KEY(3,T,S)
2960 IF S=0 THEN 2950
2970 IF T<>50 THEN 3010
2980 D=2
2990 PP=50
3000 GOTO 3280
3010 IF T<>51 THEN 3050
3020 D=3
3030 PP=51
3040 GOTO 3280
3050 IF T<>52 THEN 3090
3060 D=4
3070 PP=52
3080 GOTO 3280
3090 IF T<>53 THEN 3130
3100 D=5
3110 PP=53
3120 GOTO 3280
3130 IF T<>54 THEN 3170
3140 D=6
3150 PP=54
3160 GOTO 3280
3170 IF T<>55 THEN 3210
3180 D=7
3190 PP=55
3200 GOTO 3280
3210 IF T<>56 THEN 3250
3220 D=8
3230 PP=56
3240 GOTO 3280
3250 IF T<>57 THEN 2950
3260 D=9
3270 PP=57
3280 CALL HCHAR(23,16,PP)
3290 CALL SOUND(5,1200,0)
3300 FOR DELAI=1 TO 100
3310 NEXT DELAI
3320 GOTO 2080
3330 IF T<>83 THEN 3450
3340 CALL HCHAR(23,16,66)
3350 R=2
3360 CALL SOUND(50,550,10)
3370 FOR I=13 TO 19 STEP 2
3380 CALL VCHAR(I,CJ,32)
3390 NEXT I
3400 IF W=1 THEN 3430
3410 W=1
3420 GOTO 1840
3430 W=0
3440 GOTO 1840
3450 IF T<>78 THEN 3530
3460 CALL HCHAR(23,16,68)
3470 N=N+2
3480 R=3
3490 CALL HCHAR(Y,0,Q)
3500 CALL HCHAR(20,X,41)
3510 CALL SOUND(50,550,10)
3520 GOTO 1970
3530 IF T<>73 THEN 3550
3540 GOTO 6970
3550 IF T<>65 THEN 2080
3560 CS=0
3570 SJ=0
3580 SP=0
3590 CALL VCHAR(3,0,Q,17)
3600 CALL HCHAR(20,8,41,17)
3610 XD=0
3620 XP=0
3630 IF R=1 THEN 3650
3640 R=0
3650 N=0
3660 FOR I=15 TO 19 STEP 2
3670 CALL HCHAR(I,CJ,32)
3680 NEXT I
3690 FOR I=1 TO 6
3700 CALL HCHAR(23,16,67)
3710 CALL SOUND(20,523,10)
3720 CALL HCHAR(23,16,32)
3730 NEXT I
3740 GOTO 1970
3750 REM
3760 CALL HCHAR(Y,0,P)
3770 CS=1
3780 N=N+1
3790 CALL SOUND(10,330,10)
3800 GOTO 2080
3810 CALL HCHAR(20,X,P)
3820 CS=0
3830 N=N+1
3840 CALL HCHAR(Y,0,Q)
3850 CALL HCHAR(20,X,41)
3860 CALL SOUND(10,330,10)
3870 IF N=2 THEN 3900
3880 IF N=4 THEN 4070
3890 IF N=6 THEN 4340
3900 YD=Y
3910 XD=X
3920 CALL GCHAR(YD,XD,CD)
3930 IF W=1 THEN 3990
3940 IF CD<>120 THEN 3960
3950 GOTO 1970
3960 IF CD<>152 THEN 3560
3970 SJ=1
3980 GOTO 1970
3990 IF CD<>112 THEN 4010
4000 GOTO 1970
4010 IF CD<>153 THEN 3560
4020 SJ=1
4030 GOTO 1970
4040 IF R<>2 THEN 1970
4050 N=N+2
4060 GOTO 1970
4070 YP=Y
4080 XP=X
4090 IF R<>2 THEN 4130
4100 YN=YP
4110 XN=XP
4120 GOTO 4990
4130 CALL GCHAR(YP,XP,CP)
4140 IF W=1 THEN 4220
4150 IF CP=120 THEN 3560
4160 IF CP=152 THEN 3560
4170 IF CP<>112 THEN 4190
4180 GOTO 4280
4190 IF CP<>153 THEN 3560
```

```
4200 SP=1
4210 GOTO 4280
4220 IF CP=112 THEN 3560
4230 IF CP=153 THEN 3560
4240 IF CP<>120 THEN 4260
4250 GOTO 4280
4260 IF CP<>152 THEN 3560
4270 SP=1
4280 IF SJ=1 THEN 1970
4290 IF YP=YD-2 THEN 1970
4300 IF YP=YD+2 THEN 1970
4310 IF XP=XD+2 THEN 1970
4320 IF XP=XD-2 THEN 1970
4330 GOTO 3560
4340 YN=Y
4350 XN=X
4360 IF R<>2 THEN 4380
4370 GOTO 4990
4380 CALL GCHAR(YN,XN,CA)
4390 IF CA=112 THEN 3560
4400 IF CA=120 THEN 3560
4410 IF CA=152 THEN 3560
4420 IF CA=153 THEN 3560
4430 IF R=3 THEN 4900
4440 IF YP=YD THEN 4490
4450 IF XP=XD THEN 4530
4460 IF YP<YD THEN 4570
4470 IF YP>YD THEN 4680
4480 GOTO 3560
4490 IF YN<>YP THEN 3560
4500 IF XN=XP-2 THEN 4990
4510 IF XN=XP+2 THEN 4990
4520 GOTO 3560
4530 IF XN<>XP THEN 3560
4540 IF YN=YP-2 THEN 4990
4550 IF YN=YP+2 THEN 4990
4560 GOTO 3560
4570 IF SJ=1 THEN 4790
4580 IF XP<>XD-2 THEN 4630
4590 IF XN=XP-2 THEN 4610
4600 GOTO 3560
4610 IF YN=YP-2 THEN 4990
4620 GOTO 3560
4630 IF XP<>XD+2 THEN 3560
4640 IF XN=XP+2 THEN 4660
4650 GOTO 3560
4660 IF YN=YP-2 THEN 4990
4670 GOTO 3560
4680 IF SJ=1 THEN 4790
4690 IF XP<>XD-2 THEN 4740
4700 IF XN=XP-2 THEN 4720
4710 GOTO 3560
4720 IF YN=YP+2 THEN 4990
4730 GOTO 3560
4740 IF XP<>XD+2 THEN 3560
4750 IF XN=XP+2 THEN 4770
4760 GOTO 3560
4770 IF YN=YP+2 THEN 4990
4780 GOTO 3560
4790 IF XP>YD THEN 4850
4800 IF YN=YP-2 THEN 4820
4810 GOTO 3560
4820 IF XN=XP-2 THEN 4990
4830 IF XN=XP+2 THEN 4990
4840 GOTO 3560
4850 IF YN=YP+2 THEN 4870
4860 GOTO 3560
4870 IF XN=XP-2 THEN 4990
4880 IF XN=XP+2 THEN 4990
4890 GOTO 3560
4900 IF SJ=1 THEN 4990
4910 IF W=1 THEN 4960
4920 IF YN<>YD-2 THEN 4940
4930 GOTO 4990
4940 IF YN>=YD THEN 3560
4950 GOTO 4990
4960 IF YN<>YD+2 THEN 4980
4970 GOTO 4990
4980 IF YN<=YD THEN 3560
4990 Y=YD
5000 X=XD
5010 GOSUB 5960
5020 IF R=3 THEN 5350
5030 IF W=1 THEN 5120
5040 U1=U1-1
5050 IF U1>=48 THEN 5100
5060 U1=57
5070 IF U2=48 THEN 5100
5080 U2=U2-1
5090 CALL HCHAR(21,3,U2)
5100 CALL HCHAR(21,4,U1)
5110 GOTO 5190
5120 V1=V1-1
5130 IF V1>=48 THEN 5180
5140 V1=57
5150 IF V2=48 THEN 5180
5160 V2=V2-1
5170 CALL HCHAR(21,28,V2)
5180 CALL HCHAR(21,29,V1)
5190 IF R<>2 THEN 5210
5200 GOTO 5350
5210 Y=YP
5220 X=XP
5230 IF W=1 THEN 5290
5240 IF SJ=1 THEN 5270
5250 CALL HCHAR(Y,X,120)
5260 GOTO 5330
5270 CALL HCHAR(Y,X,152)
5280 GOTO 5330
5290 IF SJ=1 THEN 5320
5300 CALL HCHAR(Y,X,112)
5310 GOTO 5330
5320 CALL HCHAR(Y,X,153)
5330 GOSUB 5370
5340 GOSUB 5960
5350 GOSUB 5370
5360 GOTO 5390
5370 CALL SOUND(30,110,0,123,0)
5380 RETURN
5390 IF W=1 THEN 5470
5400 IF SJ=1 THEN 5450
5410 CALL HCHAR(YN,XN,120)
5420 IF YN<>3 THEN 5440
5430 CALL HCHAR(YN,XN,152)
5440 GOTO 5530
5450 CALL HCHAR(YN,XN,152)
5460 GOTO 5530
5470 IF SJ=1 THEN 5520
5480 CALL HCHAR(YN,XN,112)
5490 IF YN<>19 THEN 5510
5500 CALL HCHAR(YN,XN,153)
5510 GOTO 5530
5520 CALL HCHAR(YN,XN,153)
5530 IF U2<>48 THEN 5550
5540 IF U1=48 THEN 6830
5550 IF V2<>48 THEN 5570
5560 IF V1=48 THEN 6870
5570 IF R<>1 THEN 5730
5580 D=D-1
5590 PP=PP-1
5600 CALL SOUND(5,1200,0)
5610 FOR I=15 TO 19 STEP 2
5620 CALL VCHAR(I,CJ,32)
5630 NEXT I
```

```
5640 N=0
5650 XD=0
5660 XP=0
5670 CALL HCHAR(YN,0,Q)
5680 CALL HCHAR(20,XN,41)
5690 IF D<>0 THEN 1970
5700 R=0
5710 CALL HCHAR(23,16,32)
5720 GOTO 5790
5730 IF R<>2 THEN 5790
5740 FOR I=15 TO 19 STEP 2
5750 CALL VCHAR(I,CJ,32)
5760 NEXT I
5770 R=0
5780 GOTO 5870
5790 FOR I=13 TO 19 STEP 2
5800 CALL VCHAR(I,CJ,32)
5810 NEXT I
5820 IF W=1 THEN 5850
5830 W=1
5840 GOTO 5860
5850 W=0
5860 R=0
5870 CALL HCHAR(YN,0,Q)
5880 CALL HCHAR(20,XN,41)
5890 XD=0
5900 N=0
5910 XP=0
5920 SJ=0
5930 SP=0
5940 CALL HCHAR(23,16,32)
5950 GOTO 1840
5960 IF Y<>3 THEN 5980
5970 GOTO 6050
5980 IF Y<>7 THEN 6000
5990 GOTO 6050
6000 IF Y<>11 THEN 6020
6010 GOTO 6050
6020 IF Y<>15 THEN 6040
6030 GOTO 6050
6040 IF Y<>19 THEN 6350
6050 IF X<>10 THEN 6070
6060 GOTO 6580
6070 IF X<>14 THEN 6090
6080 GOTO 6580
6090 IF X<>18 THEN 6110
6100 GOTO 6580
6110 IF X<>22 THEN 6130
6120 GOTO 6580
6130 IF Y<>3 THEN 6190
6140 IF X<>8 THEN 6160
6150 GOTO 6560
6160 IF X<>24 THEN 6180
6170 GOTO 6780
6180 GOTO 6600
6190 IF Y<>7 THEN 6210
6200 GOTO 6420
6210 IF Y<>11 THEN 6230
6220 GOTO 6420
6230 IF Y<>15 THEN 6290
6240 IF X<>8 THEN 6260
6250 GOTO 6660
6260 IF X<>24 THEN 6280
6270 GOTO 6740
6280 GOTO 6680
6290 IF Y<>19 THEN 6350
6300 IF X<>8 THEN 6320
6310 GOTO 6700
6320 IF X<>24 THEN 6340
6330 GOTO 6740
6340 GOTO 6720
6350 IF Y<>5 THEN 6370
6360 GOTO 6470
6370 IF Y<>9 THEN 6390
6380 GOTO 6470
6390 IF Y<>13 THEN 6410
6400 GOTO 6470
6410 IF Y=17 THEN 6470
6420 IF X<>8 THEN 6440
6430 GOTO 6660
6440 IF X<>24 THEN 6460
6450 GOTO 6740
6460 GOTO 6680
6470 IF X=8 THEN 6620
6480 IF X=12 THEN 6620
6490 IF X=16 THEN 6620
6500 IF X=20 THEN 6620
6510 IF X=24 THEN 6620
6520 IF X=10 THEN 6640
6530 IF X=14 THEN 6640
6540 IF X=18 THEN 6640
6550 IF X=22 THEN 6640
6560 CALL HCHAR(Y,X,88)
6570 RETURN
6580 CALL HCHAR(Y,X,89)
6590 RETURN
6600 CALL HCHAR(Y,X,91)
6610 RETURN
6620 CALL HCHAR(Y,X,92)
6630 RETURN
6640 CALL HCHAR(Y,X,96)
6650 RETURN
6660 CALL HCHAR(Y,X,98)
6670 RETURN
6680 CALL HCHAR(Y,X,99)
6690 RETURN
6700 CALL HCHAR(Y,X,100)
6710 RETURN
6720 CALL HCHAR(Y,X,101)
6730 RETURN
6740 CALL HCHAR(Y,X,104)
6750 RETURN
6760 CALL HCHAR(Y,X,103)
6770 RETURN
6780 CALL HCHAR(Y,X,102)
6790 RETURN
6800 REM ------------------
6810 REM --   RESULTATS   --
6820 REM ------------------
6830 PW$=" JAUNES "
6840 C=11
6850 EC=2
6860 GOTO 6900
6870 PW$=" NOIRS "
6880 C=2
6890 EC=11
6900 CALL CLEAR
6910 CALL SCREEN(EC)
6920 FOR I=1 TO 7
6930 CALL COLOR(I,C,1)
6940 NEXT I
6950 PRINT TAB(3);"LES PIONS"&PW$&"SONT":
::TAB(7);"LES VAINQUEURS"::::
6960 GOTO 7060
6970 GOSUB 7000
6980 PRINT TAB(9);"PARTIE NULLE"::::
6990 GOTO 7060
7000 CALL CLEAR
7010 CALL SCREEN(14)
7020 FOR I=1 TO 12
7030 CALL COLOR(I,8,1)
7040 NEXT I
7050 RETURN
7060 FOR DEL=1 TO 800
```

```
7070 NEXT DEL
7080 CALL CLEAR
7090 PRINT TAB(4);"NOUVELLE PARTIE ?  O:N
"
7100 CALL KEY(3,T,S)
7110 IF S=0 THEN 7100
7120 IF T<>79 THEN 7160
7130 CALL CLEAR
7140 CALL SCREEN(14)
7150 GOTO 700
7160 IF T<>78 THEN 7100
7170 CALL CLEAR
7180 END
```

Galaxia

Pierre Marcellin

NDLR : Ce jeu d'arcade, écrit en ***Assembleur****, requiert un lecteur de disquette, une extension 32Ko et des manettes de jeu. Pendant le jeu, la touche ALPHA-LOCK doit être relâchée.*

Vous êtes perdu dans l'espace infini et vos réserves d'énergie ne sont pas inépuisables; il vous faut les reconstituer de temps en temps. Pour cela, vous devez prendre d'assaut les bases que défendent vos ennemis mortels : les terribles Arkoniens.

Vous êtes aux commandes du dernier produit de la technologie terrienne. Votre vaisseau est équipé de boucliers énergétiques et d'un canon-laser.

Votre bouclier pourra annuler les tirs des Arkoniens touchant votre vaisseau, mais cela demande une dépense d'énergie importante : pas moins de 100 unités. Votre canon-laser est moins gourmand puisqu'un tir utilise une unité.

Les plaques solaires qui équipent votre appareil peuvent reconstituer votre potentiel énergétique mais le rendement est très faible : une unité de temps en temps.

Quand vos réserves d'énergie sont épuisées, votre canon devient inutilisable et vous devrez attendre quer les plaques solaires fassent leur effet.

Lorsque vos réserves tombent en-dessous de 100 unités, vos boucliers ne peuvent plus arrêter la totalité d'un tir Arkonien. Dans ce cas, le tir ennemi traverse le bouclier et frappe la coque de vaisseau, provoquant aussitôt son explosion (attention, vous n'avez qu'une vie !).

Vous devez essayer de franchir le barrage formé par vos ennemis afin de les mettre en déroute. Leur technologie ne vaut pas la vôtre : ils ne connaissent pas encore le secret des boucliers énergetiques, mais attention, ils sont courageux et il faut en éliminer une bonne centaine avant de les faire fuir. De plus, si vous tardez à les mettre en déroute, ils deviendront plus hardis et se rapprocheront de votre appareil.

Quand vous les aurez fait fuir (si vous y parvenez), vous pourrez alors essayer de vous poser sur la base qu'ils défendaient, afin de récupérer un peu de cette énergie qui vous fait si cruellement défaut. La base est établie sur une planète et, bien sûr, la pesanteur se fait sentir, sans compter l'inertie de votre vaisseau. Petit problème : la base est défendue par une triple barrière énergétique assez dangereuse. Elle est composée de la même énergie que les tirs Arkoniens; votre bouclier pourra donc vous protéger, mais toujours au même prix.

De plus, il vaut mieux ne pas vous écraser sur le sol rocailleux ni contre les murs. Heureusement, l'ordinateur de bord est là. C'est un ordinateur de la dernière génération, capable d'inverser le mouvement du vaisseau une microseconde avant la catastrophe.

Cette prouesse nécessite, elle aussi, une énorme dépense d'énergie; gare à l'explosion ! De même, si vous montez trop haut, l'ordinateur annulera votre progression (cela ne nécessite que peu d'énergie, heureusement) puisque vous êtes venu ici pour vous poser, pas pour repartir.

Si vous réussissez à éviter tous ces dangers, vous pourrez vous poser sur la plate-forme prévue à cet effet. Il vous faudra détruire au moins une partie de la barrière énergétique, à l'aide de votre canon-laser, pour passer au travers sans la toucher. Attention à bien vous aligner avec la plate-forme et, surtout, à ne pas atterrir trop vite; ménagez le train d'atterissage sinon la réparation pourra vous coûter cher.

Quand vous aurez atterri, il faudra repartir par le même chemin. Il faut alors monter le plus haut possible; l'ordinateur de bord vous laissera partir mais refusera tout autre atterrissage : il n'y a plus rien à tirer de cette base.

Une fois dans l'espace, il faudra atteindre la base suivante, elle aussi défendue par des ennemis un peu plus courageux que les précédents; il faudra en éliminer deux de plus avant de les faire fuir.

Quand la partie sera terminée, à l'affichage de "FIN DU JEU", vous pourrez appuyer sur *REDO* pour une autre partie de même niveau, ou sur *BEGIN* pour recommencer au début et changer de niveau.

Le programme

Le source est subdivisé en sept parties baptisées *GALAXIA-0 à GALAXIA-6*. Au moment de l'assemblage, il faut donner *DSK1.GALAXIA-6* comme nom de fichier source, car il contient les directives *COPY*.

□

```
****************
*   GALAXIA-6   *
****************

       COPY   "DSK1.GALAXIA-0"
       COPY   "DSK1.GALAXIA-1"
       COPY   "DSK1.GALAXIA-2"
       COPY   "DSK1.GALAXIA-3"
       COPY   "DSK1.GALAXIA-4"
       COPY   "DSK1.GALAXIA-5"

       END    GALAXI
```

```
****************
*   GALAXIA-0   *
****************

        DEF GALAXI
        REF VMBW,VMBR,KSCAN,VWTR

* Dessin fusee
A       DATA 1,>102,>301,>103
        DATA >62F,>3921,>3F03,>100
        DATA >80,>8040,>C080,>80C0
        DATA >60F4,>9C84,>FCC0,>8000

* Dessin ennemis
MSTR1   DATA 0,3,>F1C,>3172
        DATA >DC8F,>100,0,0
        DATA 0,>C0,>F038,>8C4E
        DATA >3BF1,>8000,0,0

*Explosion ennemis
        DATA 0,0,>200,>40
        DATA >708,>1001,>72DC,>8F01
        DATA 0,0,0,>8000
        DATA >3009,>480,>4E3B,>F180
        DATA 0,>200,>2120,>608
        DATA >1062,>1004,>70DC,>8F01
        DATA >40,>30,>1800,>6014
        DATA >808,>802C,>8E3B,>F180
        DATA >10,>280,>4008,>200
        DATA >1400,>120,>72C0,>8720
        DATA >8002,>2400,>8000,>5000
        DATA >A008,>44,>E03,>9120
        DATA 0,>1208,>11,2
        DATA >1140,>510,4,>1000
        DATA >10,>2004,>10,>4220
        DATA >4008,>2002,>8010,0
        DATA 0,1,>401,>802
        DATA >1104,4,>100,0
        DATA 0,>40,>20,>4020
        DATA >4800,>8840,0,0
        DATA 0,0,0,>400
        DATA >0204,0,0,0
        DATA 0,0,>80,>20
        DATA >8000,>4000,0,0

* Position,code,couleur
B       DATA >AE80,>800D,>D000

* Vitesses
C       DATA >0000,>0000
D       DATA >0000

ENERGI  DATA 1000
NBENAP  DATA 0
NBVAG   DATA 107
PATRON  DATA >0400
ATTRIB  DATA >0300
VITESS  DATA >0780
FLAG    DATA 0
POINT   DATA 0
VITLIM  DATA 0
TIRLIM  DATA 0
VITBAR  DATA 0
UN      DATA 1
DEUX    DATA 2
TROIS   DATA 3
QUATRE  DATA 4
CINQ    DATA 5
SIX     DATA 6
SEPT    DATA 7
HUIT    DATA 8
NEUF    DATA 9
DIX     DATA 10
ONZE    DATA 11
TRENT2  DATA 32
UNB     BYTE 1
DEUXB   BYTE 2
H40     BYTE >40
EVEN

* UTILITAIRE ECRITURE EN VDP
        DATA 0,0
VSBW    MOV  R1,@VSBW-2
        MOV  R0,@VSBW-4
        AB   @H40,@VSBW-4
        MOVB @VSBW-3,@>8C02
        MOVB @VSBW-4,@>8C02
        MOVB @VSBW-2,@>8C00
        RT

* LECTURE EN VDP
VSBR    MOV  R0,@VSBW-4
        MOVB @VSBW-3,@>8C02
        MOVB @VSBW-4,@>8C02
        MOVB @>8800,R1
        RT

****************************
* NOMBRE ALEATOIRE DANS R1 *
****************************

RND     LI   R1,28645
        MPY  @>83C0,R1
        AI   R2,31417
        MOV  R2,@>83C0
        SWPB R1
        RT

***********************
*APPARITION D'UN ENNEMI*
***********************

VITINT  DATA 0,0
        DATA 0
ENNEMI  MOV  R11,@ENNEMI-2
        INC  @NBENAP
        C    @NBENAP,@NBVAG
        JL   ENNEM2
        BL   @TAB2
ENNEM2  MOV  @ATTRIB,R0
        LI   R1,>F000
        BL   @VSBW
        INC  R0
        BL   @RND
        BL   @VSBW
        INC  R0
        LI   R1,>8400
        BL   @VSBW
        INC  R0
ENNEM1  BL   @RND
```

```
       ANDI R1,>0F00
       CI   R1,>0100
       JLE  ENNEM1
       BL   @VSBW
       A    @QUATRE,@ATTRIB
       LI   R1,VITINT
       LI   R2,2
       MOV  @VITESS,R0
       BLWP @VMBW
       A    @QUATRE,@VITESS
       AB   @UNB,@>837A
       MOV  @ENNEMI-2,R11
       RT

**********************
*MANETTES DE JEU    *
*TESTE LES 2 MANETTES*
**********************

       DATA 0
CD     MOV  R11,@CD-2
       MOVB @UNB,@>8374
       BLWP @KSCAN
       MOV  @>8376,R1
       MOVB @DEUXB,@>8374
       BLWP @KSCAN
       MOV  @>8376,R0
       A    R0,R1
       MOV  @CD-2,R11
       RT

*****************************
*MOUVEMENT DU VAISSEAU <-> *
*****************************

       DATA 0
MVT    MOV  R11,@MVT-2
       ANDI R1,>00FF
       SLA  R1,4
       SWPB R1
       LI   R0,>781
       BL   @VSBW
       MOV  @MVT-2,R11
       RT

*******************
*DESSIN DU VAISSEAU*
*******************

       DATA 0
DESSIN MOV  R11,@DESSIN-2
       MOV  @PATRON,R0
       LI   R1,A
       LI   R2,32
       BLWP @VMBW
       MOV  @ATTRIB,R0
       LI   R1,B
       LI   R2,5
       BLWP @VMBW
       MOV  @VITESS,R0
       LI   R1,C
       LI   R2,4
       BLWP @VMBW
       LI   R0,>E200
       MOVB R0,@>83D4
       LI   R0,>01E2
       BLWP @VWTR
       LI   R3,>0100
       MOVB R3,@>837A
       A    @TRENT2,@PATRON
       A    @QUATRE,@ATTRIB
       A    @QUATRE,@VITESS
       MOV  @DESSIN-2,R11
       RT

*******************************
*MODIFICATION VITESSE ENNEMIS*
*******************************

       DATA 0
CHGT   MOV  R11,@CHGT-2
       LI   R0,>784
CHGT1  BL   @VSBR
       MOV  R1,R3
       BL   @RND
       SRA  R1,4
       A    R3,R1
       BL   @VSBW
       INC  R0
       BL   @VSBR
       MOV  R1,R3
       BL   @RND
       SRA  R1,4
       A    R3,R1
       BL   @VSBW
       AI   R0,3
       C    R0,@VITESS
       JNE  CHGT1
       MOV  @CHGT-2,R11
       RT

***************************
*CONTROLE POSITION ENNEMIS*
***************************

LIMBAS DATA >5000
       DATA 0
CTRPOS MOV  R11,@CTRPOS-2
       LI   R0,>304
CTRP1  CLR  R1
       BL   @VSBR
       CB   R1,@LIMBAS
       JL   CTRP3
       CI   R1,>E000
       JL   CTRP2
       AI   R0,>480
       BL   @VSBR
       ABS  R1
       BL   @VSBW
       AI   R0,>FB80
       B    @CTRP3
CTRP2  AI   R0,>480
       BL   @VSBR
       ABS  R1
       NEG  R1
       BL   @VSBW
       AI   R0,>FB80
CTRP3  AI   R0,4
       C    R0,@ATTRIB
       JNE  CTRP1
       MOV  @CTRPOS-2,R11
       RT

***************************
*CONTROLE VITESSES ENNEMIS*
***************************

       DATA 0
CTRVIT MOV  R11,@CTRVIT-2
       LI   R0,>784
CTRVIA BL   @CTRV1
       INC  R0
       BL   @CTRV1
       AI   R0,3
```

```
       C      R0,@VITESS
       JNE    CTRVIA
       MOV    @CTRVIT-2,R11
       RT

       DATA   0
CTRV1  MOV    R11,@CTRV1-2
       BL     @VSBR
       MOV    R1,R9
       ABS    R1
       C      R1,@VITLIM
       JL     CTRV3
       MOV    @VITLIM,R3
       MOV    R9,R9
       JGT    CTRV2
       NEG    R3
CTRV2  MOV    R3,R1
       BL     @VSBW
CTRV3  MOV    @CTRV1-2,R11
       RT

FLGEXP DATA   0
FLGTE  DATA   0
HA000  DATA   >A000
H2000  DATA   >2000
H8000  DATA   >8000

*******************
*DEBUT DU PROGRAMME*
*******************

GALAXI BL     @TITRE
LUTIN  LI     R0,>07F1
       BLWP   @VWTR
       BL     @HCHAR
       DATA   >380,>F100,18
       BL     @HCHAR
       DATA   >392,>9100,15
       LI     R0,>391
       LI     R1,>B100
       BL     @VSBW
       BL     @DESSIN
       BL     @CP
       BL     @ENNEMI
       BL     @ENNEMI
       BL     @ENNEMI
       BL     @ENNEMI
       BL     @ENNEMI
       BL     @ENNEMI
       BL     @ENNEMI
       MOV    @ATTRIB,R0
       LI     R1,>D000
       BLWP   @VSBW
KEY    CB     @LIMBAS,@HA000
       JHE    LUTIN4
       INC    @LIMBAS
LUTIN4 A      @H100,@FLGENG
       JNE    LUTIN7
       INC    @ENERGI
LUTIN7 BL     @CD
       BL     @MVT
       BL     @CTRPOS
       A      @H2000,@FLGEXP
       JNE    LUTIN6
       BL     @EXP
       JMP    LUTIN1
LUTIN6 BL     @CHGT
LUTIN1 BL     @TIR
       BL     @CTRLTV
       BL     @TIRE
       A      @H8000,@FLGTE
       JNE    LUTIN5
```

```
       BL     @DEPLTE
       JMP    LUTIN3
LUTIN5 BL     @AFFPTS
       BL     @CTRVIT
LUTIN3 BL     @DEPLT
       BL     @CTRLTE
       LIMI   2
       LIMI   0
       B      @KEY


****************
*   GALAXIA-1  *
****************

* PATRONS DES TIRS

CRCTR1 BYTE   >80,>80,>80,>80,>80,>80
       BYTE   >80,>80
       BYTE   >40,>40,>40,>40,>40,>40
       BYTE   >40,>40
       BYTE   >20,>20,>20,>20,>20,>20
       BYTE   >20,>20
       BYTE   >10,>10,>10,>10,>10,>10
       BYTE   >10,>10
       BYTE   8,8,8,8,8,8,8,8
       BYTE   4,4,4,4,4,4,4,4
       BYTE   2,2,2,2,2,2,2,2
       BYTE   1,1,1,1,1,1,1,1
       EVEN
CARACT DATA   0
NBTV   DATA   0
POSTV  DATA   0
X      DATA   0
Y      DATA   0

**********************
*CHARGEMENT DES PATRONS*
**********************

CP     LI     R0,>C08
       LI     R1,CRCTR1
       LI     R2,64
       BLWP   @VMBW
       LI     R0,>420
       LI     R1,MSTR1
       LI     R2,32+192
       BLWP   @VMBW
       LI     R0,>C78
       LI     R1,SOL
       LI     R2,184
       BLWP   @VMBW
       RT

************
*TIR VAISSEAU*
************

B12    DATA   >1200
D22    DATA   22
       DATA   0
TIR    MOV    R11,@TIR-2
       MOV    @ENERGI,@ENERGI
       JEQ    TIR1
       MOV    @NBTV,R0
       JEQ    TIR2
TIR1   MOV    @TIR-2,R11
       RT
TIR2   MOVB   @UNB,@>8374
       BLWP   @KSCAN
       CB     @>8375,@B12
```

```
       JEQ   TIR3
       MOVB  @DEUXB,@>8374
       BLWP  @KSCAN
       CB    @>8375,@B12
       JNE   TIR1
TIR3   DEC   @ENERGI
       LI    R0,>301
       CLR   R1
       BL    @VSBR
       SWPB  R1
       AI    R1,7
       MOV   R1,@X
       MOV   @D22,@Y
       BL    @TIRA
       JMP   TIR1

XA     DATA  0
       DATA  0
TIRA   MOV   R11,@TIRA-2
       INC   @NBTV
       MOV   @X,R1
       SRA   R1,3
       MOV   R1,@XA
       INC   @XA
       SLA   R1,3
       MOV   @X,R6
       S     R1,R6
       MOV   R6,R1
       MOV   @Y,R0
       SLA   R0,5
       DEC   R0
       A     @XA,R0
       AI    R1,128
       SWPB  R1
       BL    @VSBW
       MOV   R0,@POSTV
       MOV   R1,@CARACT
       MOV   @TIRA-2,R11
       RT

*****************
*BOUCLE D'ATTENTE*
*****************

DELAI  CLR   R15
       INC   R15
       JNE   DELAI+2

       RT

*****************************
*DEPLACEMENT DU TIR VAISSEAU*
*****************************

H0     DATA  0
       DATA  0
DEPLT  MOV   R11,@DEPLT-2
       C     @NBTV,@H0
       JEQ   DEPLT3
DEPLT1 MOV   @POSTV,R0
       LI    R1,>2000
       BL    @VSBW
       AI    R0,-32
       JGT   DEPLT2
       CI    R0,-32
       JL    DEPLT2
       CLR   @NBTV
       JMP   DEPLT3
DEPLT2 MOV   @CARACT,R1
       BL    @VSBW
       MOV   R0,@POSTV
DEPLT3 MOV   @DEPLT-2,R11
       RT

************************
*CONTROLE TIRS VAISSEAU*
************************

NBED   DATA  >320
ADRATT DATA  0
BUFFER BSS   10
BUF1   BSS   4
BD0    DATA  >D000
NUNB   DATA  -1
HN32   DATA  -32
HN4    DATA  -4
B88    DATA  >8800

       DATA  0
CTRLTV MOV   R11,@CTRLTV-2
       C     @NBTV,@H0
       JNE   CTRTV6
       B     @CTRTV3
CTRTV6 LI    R0,>304
       MOV   R0,@ADRATT
CTRTV2 CLR   R1
       BL    @VSBR
       MOV   R1,R3
       SWPB  R3
       INC   R0
       BL    @VSBR
       SWPB  R1
       MOV   R1,R4
       AI    R3,10

CTRTV4 MOV   @POSTV,R6
       MOV   R6,R0
       MOV   R6,R9
       SRA   R6,5
       MOV   R6,R8
       SLA   R8,3
       SLA   R6,5
       S     R6,R9
       SLA   R9,3
       CLR   R1
       BL    @VSBR
       SWPB  R1
       AI    R1,-128
       A     R1,R9
       C     R9,R4
       JL    CTRTV5
       AI    R4,15
       C     R4,R9
       JL    CTRTV5
       C     R3,R8
       JL    CTRTV5
       AI    R8,7
       AI    R3,-7
       C     R8,R3
       JL    CTRTV5


     ****************
     *ENNEMI DETRUIT*
     ****************

       INC   @POINT
       BL    @SEEN
       MOV   @POSTV,R0
       CLR   @NBTV
       LI    R1,>2000
       BL    @VSBW
       MOV   @ADRATT,R0
       LI    R1,BUFFER
       LI    R2,4
       BLWP  @VMBR
```

```
       MOVB @BD0,@BUFFER+4
       AI   R0,>480
       LI   R1,BUF1
       DECT R2
       BLWP @VMBR
       MOVB @B88,@BUFFER+2
       MOV  @NBED,R0
       LI   R1,BUFFER
       LI   R2,5
       BLWP @VMBW
       AI   R0,>480
       LI   R1,BUF1
       LI   R2,2
       BLWP @VMBW

***************************
*Transfert dernier sprite *
***************************

       A    @HN4,@ATTRIB
       MOV  @ATTRIB,R0
       MOV  @ADRATT,R3
       A    @HN4,@VITESS
       BL   @TRSFRT
       A    @QUATRE,@NBED
       A    @HN32,@PATRON
       BL   @ENNEMI
       JMP  CTRTV3
CTRTV5 A    @QUATRE,@ADRATT
       MOV  @ADRATT,R0
       C    R0,@ATTRIB
       JHE  CTRTV3
       B    @CTRTV2
CTRTV3 MOV  @CTRLTV-2,R11
       RT

************************
*EXPLOSION D'UN ENNEMIS*
************************

       DATA 0
EXP    MOV  R11,@EXP-2
       MOV  @ATTRIB,R0
EXP5   INCT R0
EXP1   C    R0,@NBED
       JL   EXP3
EXP2   MOV  @EXP-2,R11
       RT
EXP3   CLR  R1
       BL   @VSBR
       CI   R1,>9C00
       JHE  EXP4
       AI   R1,>400
       BL   @VSBW
       AI   R0,4
       JMP  EXP1

EXP4   DECT R0
       MOV  R0,R3
       MOV  @NBED,R0
       AI   R0,-4
       BL   @TRSFRT
       A    @HN4,@NBED
       AB   @NUNB,@>837A
       MOV  @NBED,R0
       LI   R1,>D000
       BL   @VSBW
       B    @EXP5

*********************
*TRANSFERT DE SPRITE*
*********************
```

```
AR0    DATA 0
TRS    BSS  8

       DATA 0
TRSFRT MOV  R11,@TRSFRT-2
       MOV  R0,@AR0
       LI   R1,TRS
       LI   R2,4
       BLWP @VMBR
       MOV  R3,R0
       BLWP @VMBW
       MOV  @AR0,R0
       AI   R0,>480
       DECT R2
       BLWP @VMBR
       MOV  R3,R0
       AI   R0,>480
       BLWP @VMBW
       MOV  @AR0,R0
       MOV  @TRSFRT-2,R11
       RT
```

```
****************
*   GALAXIA-2  *
****************

* TIR ENNEMI

NBTE   DATA 0
POSTE  BSS  40
CARCTE BSS  40
D40    DATA 40

       DATA 0
TIRE   MOV  R11,@TIRE-2
       C    @NBTE,@D40
       JHE  TIRE1
       BL   @RND
       C    R1,@TIRLIM
       JLE  TIRE2
TIRE1  MOV  @TIRE-2,R11
       RT
TIRE2  BL   @RND
       SWPB R1
       ANDI R1,>0007
       JNE  TIRE3
       INC  R1
TIRE3  LI   R0,>300
       SLA  R1,2
       A    R1,R0
       JMP  TIRE4
TIRE5  MOV  R11,@TIRE-2
       LI   R0,>300
       CLR  @NBTE
TIRE4  CLR  R1
       BL   @VSBR
       MOV  R1,R3
       SWPB R3
       AI   R3,11
       INC  R0
       BL   @VSBR
       MOV  R1,R4
       SWPB R4
       AI   R4,8
       CI   R3,>BE
       JH   TIRE1
       MOV  R4,R5
       DEC  R3
       SRA  R3,3
```

```
       DEC   R4
       SRA   R4,3
       MOV   R4,R6
       SLA   R6,3
       S     R6,R5
       SLA   R3,5
       A     R4,R3
       MOV   R3,R0
       MOV   @NBTE,R2
       MOV   R0,@POSTE(R2)
       INCT  @NBTE
       MOV   R5,R1
       AI    R1,128
       SWPB  R1
       BL    @VSBW
       MOV   R1,@CARCTE(R2)
       B     @TIRE1

**************************
*DEPLACEMENT TIRS ENNEMIS*
**************************

       DATA  0
DEPLTE MOV   R11,@DEPLTE-2
       CLR   R2
       CLR   R1
DEPLE4 C     R2,@NBTE
       JHE   DEPLE1
       MOV   @POSTE(R2),R0
       LI    R1,>2000
       BL    @VSBW
       AI    R0,32
       CI    R0,767
       JH    DEPLE3
       MOV   @CARCTE(R2),R1
DEPLE9 BL    @VSBW
       MOV   R0,@POSTE(R2)
       INCT  R2
       B     @DEPLE4
DEPLE3 DECT  @NBTE
       MOV   @NBTE,R9
       MOV   @POSTE(R9),@POSTE(R2)
       MOV   @CARCTE(R9),@CARCTE(R2)
       B     @DEPLE4
DEPLE1 MOV   @DEPLTE-2,R11
       RT

**********************
*CONTROLE TIRS ENNEMIS*
**********************

FIN    TEXT  'FIN DU JEU'

* PATRONS EXPLOSION VAISSEAU
EXPV   DATA  1,>4132,>304,>810
       DATA  >128,>3125,>3004,>4100
       DATA  >80,>8244,>C820,>900B
       DATA  >854,>C04,>4C20,>8402
       DATA  33,>4102,>804,4096
       DATA  4132,>3021,>3008,640
       DATA  >80,>8422,>800,>9403
       DATA  >A4,>C04,>2008,>2182
       DATA  >8043,>400,4128,512
       DATA  >4024,288,264,>2200
       DATA  65,>1288,>8400,>1201
       DATA  >800A,544,>408,>2081
       DATA  0,>821,4,80
       DATA  >1002,72,0,1024
       DATA  32,64,48,>82
       DATA  >820,>8010,>4A0,0
       DATA  0,>801,0,64
       DATA  2,32,0,>400
       DATA  32,0,8,>82
       DATA  >800,>8000,>420,0
       DATA  0,0,8,0,0,32,0,0
       DATA  0,>4000,0,>480,0,0,>2000
       DATA  0,0,0,0,0,0,0,0,0,0,0,0
       DATA  0,0,0,0,0

DN100  DATA  -100
       DATA  0

CTRLTE MOV   R11,@CTRLTE-2
       CLR   R2
       LI    R0,>300
       CLR   R1
       BL    @VSBR
       MOV   R1,R6
       INC   R0
       BL    @VSBR
       MOV   R1,R7
       SWPB  R6
       SWPB  R7
       INCT  R7
       INC   R6
CTRTE1 C     R2,@NBTE
       JHE   CTRTE2
       MOV   @POSTE(R2),R4
       MOV   R4,R5
       SRA   R4,5
       MOV   R4,R0
       SLA   R4,3
       INC   R4
       SLA   R0,5
       S     R0,R5
       SLA   R5,3
       MOV   @CARCTE(R2),R0
       SWPB  R0
       AI    R0,-128
       A     R0,R5
       MOV   R6,R8
       MOV   R7,R9
       AI    R8,7
       C     R4,R8
       JLE   CTRTE4
       C     R5,R9
       JLE   CTRTE4
       AI    R9,12
       C     R9,R5
       JLE   CTRTE4

********
*TOUCHE*
********

       DECT  @NBTE
       MOV   @POSTE(R2),R0
       LI    R1,>2000
       BL    @VSBW

       MOV   @NBTE,R9
       MOV   @POSTE(R9),@POSTE(R2)
       MOV   @CARCTE(R9),@CARCTE(R2)

********
*ALARME*
********

       BL    @SONEXP
       A     @DN100,@ENERGI
       JGT   CTRTE2

****************
*PLUS D'ENERGIE*
****************
```

```
       BL    @EFFAC
       BL    @AFFPTS
CTRTE7 BL    @HCHAR
       DATA  >780,0,56

**********************
*EXPLOSION DU VAISSEAU*
**********************

       LI    R1,EXPV
CTRTE6 CI    R1,EXPV+192
       JH    CTRTE5
       LI    R0,>400
       LI    R2,32
       BLWP  @VMBW
       BL    @EXPDEL
       AI    R1,32
       JMP   CTRTE6
CTRTE5 BL    @DELAI
       BL    @DELAI
       CLR   @NBTE
       CLR   @NBTV
       LI    R0,363
       LI    R1,FIN
       LI    R2,10
       BLWP  @VMBW
       B     @FINJEU

***************
*SI PAS TOUCHE*
***************

CTRTE4 INCT  R2
       B     @CTRTE1
CTRTE2 MOV   @CTRLTE-2,R11
       RT

*****************
*BOUCLE D'ATTENTE*
*****************

EXPDEL LI    R15,>9000
       LIMI  2
EXPDE1 INC   R15
       JNE   EXPDE1
       LIMI  0
       RT

*************
*->CALL CLEAR*
*************

       DATA  0
EFFAC  MOV.  R11,@EFFAC-2
       BL    @HCHAR
       DATA  0,>2000,768
       MOV   @EFFAC-2,R11
       RT

*******************
*UTILITAIRE D'ECRAN*
*******************

       DATA  0
HCHAR  MOV   *R11+,R0
       MOV   *R11+,R1
       MOV   *R11+,R2
       MOV   R11,@HCHAR-2
HCHAR1 BL    @VSBW
       INC   R0
       DEC   R2
       JNE   HCHAR1
```

```
       MOV   @HCHAR-2,R11
       RT

VCHAR  MOV   *R11+,R0
       MOV   *R11+,R1
       MOV   *R11+,R2
       MOV   R11,@HCHAR-2
VCHAR1 BL    @VSBW
       AI    R0,32
       DEC   R2
       JNE   VCHAR1
       MOV   @HCHAR-2,R11
       RT

*********************
*AFFICHAGE DES POINTS*
*********************

ENERG  TEXT  'ENERGIE:'
SCORE  TEXT  'SCORE:'
REGAFF BSS   5
       EVEN

       DATA  0
AFFPTS MOV   R11,@AFFPTS-2
       MOV   @POINT,R0
       BL    @CVRSN
       LI    R0,2
       LI    R1,SCORE
       LI    R2,11
       BLWP  @VMBW
       MOV   @ENERGI,R0
       BL    @CVRSN
       LI    R0,19
       LI    R1,ENERG
       LI    R2,7
       BLWP  @VMBW
       LI    R0,27
       LI    R1,REGAFF+1
       LI    R2,4
       BLWP  @VMBW
       MOV   @AFFPTS-2,R11
       RT

       DATA  0
CVRSN  MOV   R11,@CVRSN-2
       CLR   R3
       LI    R1,-10000
       BL    @DECPT
       LI    R1,-1000
       INC   R3
       BL    @DECPT
       LI    R1,-100
       INC   R3
       BL    @DECPT
       LI    R1,-10
       INC   R3
       BL    @DECPT
       SETO  R1
       INC   R3
       BL    @DECPT
       MOV   @CVRSN-2,R11
       RT

       DATA  0
DECPT  MOV   R11,@DECPT-2
       SETO  R2
DECPT1 INC   R2
       A     R1,R0
       JLT   DECPT2
       JMP   DECPT1
DECPT2 AI    R2,48
```

```
       SWPB R2
       MOVB R2,@REGAFF(R3)
       S    R1,R0
       MOV  @DECPT-2,R11
       RT

****************************************
* ALARME QUAND LE VAISSEAU EST TOUCHE *
****************************************

SON1   BYTE 5,>9F,>BF,>DF,>FF,>E3,1
       BYTE 5,>89,>3F,>E2,>90,>F0,10
       BYTE 5,>89,>3F,>E2,>91,>F1,10
       BYTE 5,>89,>3F,>E2,>92,>F2,10
       BYTE 5,>89,>3F,>E2,>93,>F3,10
       BYTE 5,>89,>3F,>E2,>94,>F4,10
       BYTE 5,>89,>3F,>E2,>95,>F5,10
       BYTE 5,>89,>3F,>E2,>96,>F6,10
       BYTE 5,>89,>3F,>E2,>97,>F7,10
       BYTE 5,>89,>3F,>E2,>98,>F8,10
       BYTE 5,>89,>3F,>E2,>99,>F9,10
       BYTE 5,>89,>3F,>E2,>9A,>FA,10
       BYTE 5,>89,>3F,>E2,>9B,>FB,10
       BYTE 5,>89,>3F,>E2,>9C,>FC,10
       BYTE 5,>89,>3F,>E2,>9D,>FD,10
       BYTE 5,>89,>3F,>E2,>9E,>FE,10
       BYTE 2,>FF,>9F,0
       EVEN

SONEXP LI   R1,SON1
       LI   R0,>1000
       LI   R2,116
       BLWP @VMBW

       LI   R10,>1000
       MOV  R10,@>83CC
       SOCB @UNB,@>83FD
       MOVB @UNB,@>83CE
       RT

***************************
*BOUCLE D'ATTENTE VARIABLE*
***************************

DELVAR MOV  *R11+,R15
DELVA1 DEC  R15
       JNE  DELVA1
       RT

************
*FIN DU JEU*
************

HISCOR DATA 0
THISC  TEXT 'MEILLEUR SCORE :
FINJEU C    @POINT,@HISCOR
       JL   FIN1
       MOV  @POINT,@HISCOR
FIN1   LI   R0,100
       LI   R1,THISC
       LI   R2,17
       BLWP @VMBW
       MOV  @HISCOR,R0
       BL   @CVRSN
       LI   R0,117
       LI   R1,REGAFF
       LI   R2,5
       BLWP @VMBW

*******************************
* MISE A JOUR DES COMPTEURS *
* POUR UNE AUTRE PARTIE      *
*******************************
       CLR  @POINT
       LI   R0,1000
       MOV  R0,@ENERGI
       LIMI 2
       BL   @DELAI
       BL   @DELVAR
       DATA >5000
       LIMI 0

FIN2   CLR  R0
       LIMI 2
       LIMI 0
       CLR  @>8374
       BLWP @KSCAN
       MOVB @>8375,R0
       CI   R0,>FF00
       JEQ  FIN2
       CI   R0,>600
       JEQ  REDO

       CI   R0,>E00
       JNE  FIN2

       BL   @TITRE
REDO   BL   @DELAI
       BL   @DELAI
       BL   @DELAI
       LI   R0,105
       MOV  R0,@NBVAG
       B    @TAB3
```

```
****************
*  GALAXIA-3   *
****************

*DESSIN BASE
FLGENG DATA 0
H100   DATA 100
DN1    DATA -1
ATTAB2 BYTE -20,96,>80,>0D,>D0
VITAB2 BYTE 4,0
FLGROT DATA 0
DXYZ   DATA 0
       EVEN

********************
* PATRONS VAISSEAU *
********************

PATAB2 DATA 0,0,>102,>102
       DATA >2F39,>3F01,0,0
       DATA 0,0,>8040,>8040
       DATA >F49C,>FC80,0,0
       DATA 0,1,>107,>101
       DATA >204,>77F,>1010,0
       DATA 0,0,>C0,0
       DATA >8040,>C0FC,>1010,0

******************************
* PATRONS CARACTERES DU SOL *
******************************

SOL    DATA >7EC3,>A599,>99A5,>C37E
       DATA >E0F8,>FCFF,>FFFF,>FFFF
       DATA 0,>38FC,>FFFF,>FFFF
       DATA 0,0,>F0FC,>FEFF
       DATA >FFFE,>FCF8,>F8F8,>FCFF
       DATA >FF7F,>7F3F,>3F3F,>7FFF
       DATA >E,>3F3F,>7F7F,>FFFF
       DATA 0,>C0F3,>FFFF,>FFFF
       DATA >FFFC,>F0E0,>E0F0,>FCFF
```

```
        DATA >FF7F,>1F1F,>3F3F,>7FFF
        DATA >FFFF,>FFFF,>FFFF,>FFFF
        DATA >FFFF,>FF7F,>7F1F,>F0F
        DATA >FFFF,>FEFC,>FCFE,>FEFC
        DATA >F0F0,>E0C0,>C0C0,>A0A0
        DATA >F07,>703,>307,>70F
        DATA >F0F,>707,>703,>303
        DATA >FCFC,>F8F8,>F8F0,>F0F0
        DATA >103,>1F3F,>FFFF,>FFFF
        DATA 0,0,>307,>1FFF
        DATA >E0F0,>F0F0,>F8F8,>FCFC
        DATA >303,>707,>703,>303
        DATA >F0E0,>E0F0,>F8F8,>FEFF
        DATA >303.>707,>303,>4FFF

*******************
* DESSIN DE LA BASE *
*******************

MUR     DATA 353,>90,354,>91,355,>92
        DATA 381,>95,382,>96,383,>91
        DATA 419,>93,483,>97,547,>9B
        DATA 579,>9F,611,>9C,445,>94
        DATA 509,>98,573,>9A,605,>9D
        DATA 637,>9D,669,>9D,701,>9E
        DATA 643,>A2,675,>9F,0
MUR1    BYTE >A4,>A1,>91,>96,>A0,>99,>99
        BYTE >91,>A0,>90,>91,>92,>A1,>92
        BYTE >A1,>91,>96,>A0,>90,>91,>96
        BYTE >91,>92,>A1,>96,>92,>A5

************
* TABLEAU 2 *
************

TAB2    BL   @EFFAC
        BL   @AFFPTS
        LI   R0,>304
        LI   R1,>D000
        BL   @VSBW
        MOVB @UNB,@>837A
        BL   @HCHAR
        DATA >780,0,4
        LIMI 2
        BL   @DELAI
        BL   @DELAI
        BL   @DELAI
        LIMI 0
        CLR  @NBTV
        MOVB @H0,@>837A
        LI   R0,>300
        LI   R1,>D000
        BL   @VSBW
        BL   @DELAI

******************
* DESSINE LA BASE *
******************

        LI   R0,707
        LI   R1,MUR1
        LI   R2,27
        BLWP @VMBW
        LI   R0,>391
        LI   R1,>A100
        BL   @VSBW
        BL   @HCHAR
        DATA >392,>9100,2
        BL   @VCHAR
        DATA 352,>9900,13
        BL   @VCHAR
        DATA 385,>9900,12
        BL   @VCHAR
        DATA 386,>9900,12
        BL   @VCHAR
        DATA 414,>9900,12
        BL   @VCHAR
        DATA 415,>9900,12
        BL   @HCHAR
        DATA 739,>9900,27
        BL   @HCHAR
        DATA 387,>8F00,27
        BL   @HCHAR
        DATA 451,>8F00,27
        BL   @HCHAR
        DATA 515,>8F00,27
TAB22   MOV  @MUR(R2),R0
        JEQ  TAB21
        INCT R2
        MOV  @MUR(R2),R1
        INCT R2
        SWPB R1
        BL   @VSBW
        JMP  TAB22

**********************
* DESSINNE LE VAISSEAU *
**********************

TAB21   BL   @DELAI
        LI   R0,>400
        LI   R1,A
        LI   R2,32
        BLWP @VMBW
        LI   R0,>300
        LI   R1,ATTAB2
        LI   R2,5
        BLWP @VMBW
        LI   R0,>780
        LI   R1,VITAB2
        LI   R2,2
        BLWP @VMBW
        MOVB @UNB,@>837A
        LIMI 2
        BL   @DELAI
        BL   @DELAI
        LIMI 0
        LI   R0,>400
        LI   R1,PATAB2
        LI   R2,32
        BLWP @VMBW
        BL   @DELVAR
        DATA >5000
        LI   R0,>400
        LI   R2,32
        AI   R1,32
        BLWP @VMBW
        CLR  @NBTE
TAB23   MOV  @DXYZ,@DXYZ
        JEQ  TXYZ0
        DEC  @DXYZ
        JMP  TXYZ
TXYZ0   BL   @DEPL2
TXYZ    BL   @TIRV2
        BL   @CTIR2
        BL   @AFFPTS
        BL   @CTRPV2
        BL   @CBE
        A    @H100,@FLGENG
        JNE  TAB25
        INC  @ENERGI
TAB25   A    @VITBAR,@FLGROT
        JNE  TAB26
        BL   @ROTAT
```

```
TAB26   C      @ENERGI,@DN1
        JGT    TAB24
        BL     @SONEXP
        B      @CTRTE7
TAB24   LIMI   2
        LIMI   0
        JMP    TAB23

**************
*TIR VAISSEAU*
**************

        DATA 0
TIRV2   MOV    R11,@TIRV2-2
        C      @NBTE,@H0
        JNE    TIRV21
        C      @ENERGI.,@UN
        JLT    TIRV21
        MOVB   @UNB,@>8374
        BLWP   @KSCAN
        CB     @>8375,@B12
        JEQ    TIRV22
        MOVB   @DEUXB,@>8374
        BLWP   @KSCAN
        CB     @>8375,@B12
        JNE    TIRV21
TIRV22  DEC    @ENERGI
        BL     @TIRE5
        INC    @NBTE
TIRV21  MOV    @TIRV2-2,R11
        RT

**********************
*DEPLACEMENT VAISSEAU*
**********************

H4000   DATA >4000
FLGDEP  DATA 0
        DATA 0
DEPL2   MOV    R11,@DEPL2-2
        BL     @CD
DEPL5   MOV    R1,R5
        JEQ    DEXYZ
        LI     R6,15
        MOV    R6,@DXYZ
DEXYZ   MOV    R1,R6
        ANDI   R5,>FF00
        ANDI   R6,>FF
        SWPB   R6
        LI     R0,>780
        CLR    R1
        BL     @VSBR
        S      R5,R1
        A      @H4000,@FLGDEP
        JNE    DEPL6
        AI     R1,>100
DEPL6   MOV    R1,R9
        ABS    R9
        CI     R9,>7C00
        JHE    DEPL3

        BL     @VSBW
DEPL3   INC    R0
        CLR    R1
        BL     @VSBR
        A      R6,R1
        MOV    R1,R9
        ABS    R9
        CI     R9,>4000
        JHE    DEPL4
        BL     @VSBW
DEPL4   MOV    @DEPL2-2,R11
        RT

****************************
* CONTROLE,DEPLACEMENT TIR *
****************************

        DATA 0
CTIR2   MOV    R11,@CTIR2-2
        C      @NBTE,@H0
        JEQ    CTIR21
        MOV    @POSTE,R0
        AI     R0,32
        CLR    R1
        BL     @VSBR
        CI     R1,>2000
        JNE    CTIR22
        AI     R0,-32
        BL     @VSBR
        CI     R1,>8F00
        JEQ    CTIR25
        LI     R1,>2000
        BL     @VSBW
CTIR25  AI     R0,32
        MOV    @CARCTE,R1
        BL     @VSBW
        MOV    R0,@POSTE
CTIR21  MOV    @CTIR2-2,R11
        RT
CTIR22  CI     R1,>8F00
        JNE    CTIR23

*******************
* BARRIERE TOUCHEE *
*******************

        LI     R1,>2000
        BL     @VSBW
CTIR24  AI     R0,-32
        BL     @VSBW
        CLR    @NBTE
        JMP    CTIR21
CTIR23  LI     R1,>2000
        JMP    CTIR24

*********************************
*ROTATION BARRIERE ENERGETIQUE*
*********************************

        DATA 0
ROTAT   MOV    R11,@ROTAT-2
        LI     R0,387
        LI     R2,1
        LI     R4,413
        BL     @ROT1
        LI     R0,477
        SETO   R2
        LI     R4,451 .
        BL     @ROT1
        LI     R0,515
        LI     R2,1
        LI     R4,541
        BL     @ROT1
        MOV    @ROTAT-2,R11
        RT

        DATA 0
ROT1    MOV    R11,@ROT1-2
        BL     @VSBR
        MOV    R1,R3
        CLR    R1
ROT2    A      R2,R0
        BL     @VSBR
        CI     R1,>8000
        JL     ROT3
```

```
       CI    R1,>8900
       JL    ROT2
ROT3   S     R2,R0
       BL    @VSBW
       A     R2,R0
       C     R0,R4
       JNE   ROT2
       MOV   R3,R1
       BL    @VSBW
       MOV   @ROT1-2,R11
       RT

****************************
*CONTROLE POSITION VAISSEAU*
****************************

DRATT  DATA 0
       DATA 0
CTRPV2 MOV   R11,@CTRPV2-2
       CLR   R8
       LI    R0,>300
       CLR   R1
       BL    @VSBR
       CI    R1,>4800
       JLE   CTPV2A
       CI    R1,>A000
       JLE   CTPV22
       INC   R0
       BL    @VSBR
       CI    R1,>3C00
       JLE   CTPV23
       CI    R1,>4300
       JLE   CTPV24

**************
*touche le sol*
**************

CTPV23 LI    R0,>780
       LI    R1,>F900
       BL    @VSBW
       SETO  R8
       JMP   CTPV29

*************
*atterissage*
*************

CTPV24 MOV   @DRATT,@DRATT
       JNE   CTPV23
       BL    @DELAI
       B     @ATTER

*************************************
* VERIFICATION SORTIE PAR LE HAUT *
*************************************

CTPV2A CI    R1,>800
       JH    CTPV21
       MOV   @DRATT,@DRATT
       JEQ   CTPV2B
       B     @TAB3
CTPV2B LI    R1,>F00
       LI    R0,>780
       BL    @VSBW
       DECT  @ENERGI
       JMP   CTPV21
CTPV22 LI    R0,>301
       BL    @VSBR
       CI    R1,>1800
       JL    CTPV23
       CI    R1,>E000
       JH    CTPV23
CTPV29 LI    R0,>301
       BL    @VSBR
       CI    R1,>2000
       JLE   CTPV25
       CI    R1,>D800
       JLE   CTPV21
       JMP   CTPV26
CTPV25 LI    R1,>0600
       JMP   CTPV27
CTPV26 LI    R1,>F900
CTPV27 LI    R0,>781
       BL    @VSBW
       SETO  R8
CTPV21 CI    R8,>FFFF
       JNE   CTPV28
       A     @DN100,@ENERGI
CTPV28 MOV   @CTRPV2-2,R11
       RT

***************************
*COLLISION AVEC LA BARRIERE*
***************************

COINC  DATA 0,1,1,30,1,1,30,1,1
DIRX   DATA 0,8,8,-16,8,8,-16,8,8
DIRY   DATA 0,0,0,8,0,0,8,0,0
DN50   DATA -50
ATT    BYTE >84,>99,>D0
       EVEN
       DATA 0
CBE    MOV   R11,@CBE-2
       LI    R0,>300
       CLR   R1
       BL    @VSBR
       SWPB  R1
       SRA   R1,3
       MOV   R1,R4
       SLA   R4,3
       DEC   R4
       SLA   R1,5
       MOV   R1,R7
       CLR   R1
       INC   R0
       BL    @VSBR
       SWPB  R1
       DEC   R1
       SRA   R1,3
       MOV   R1,R5
       SLA   R5,3
       AI    R5,32
       A     R1,R7
       LI    R0,>420
       LI    R1,SOL
       LI    R2,8
       BLWP  @VMBW
       BL    @HCHAR
       DATA  >428,0,24
       CLR   R6
CBE1   A     @COINC(R6),R7
       A     @DIRX(R6),R5
       A     @DIRY(R6),R4
       INCT  R6
       CI    R6,19
       JH    CBE2
       MOV   R7,R0
       CLR   R1
       BL    @VSBR
       CI    R1,>8F00
       JNE   CBE1
       LI    R0,>304
       MOV   R4,R1
```

```
       SWPB R1
       BL   @VSBW
       INC  R0
       MOV  R5,R1
       SWPB R1
       BL   @VSBW
       INC  R0
       LI   R1,ATT
       LI   R2,3
       BLWP @VMBW
       MOVB @DEUXB,@>837A
       CLR  R9
       MOVB @>837B,R9

       MOVB @UNB,@>837A
       ANDI R9,>2000
       JEQ  CBE1
       BL   @SEEN
       A    @DN50,@ENERGI
CBE2   LI   R0,>304
       LI   R1,>D000
       BL   @VSBW
       MOV  @CBE-2,R11
       RT

***************************
*RETOUR AU PREMIER TABLEAU*
***************************

H400   DATA >400
H300   DATA >300
H780   DATA >780
H320   DATA >320

TAB3   INCT @NBVAG
       CLR  @NBENAP
       MOV  @H400,@PATRON
       MOV  @H300,@ATTRIB
       MOV  @H780,@VITESS
       LI   R0,>5000
       MOV  R0,@LIMBAS
TAB31  MOV  @H320,@NBED
       CLR  @NBTE
       CLR  @DRATT
       BL   @EFFAC
       BL   @AFFPTS
       BL   @HCHAR
       DATA >420,0,256
       B    @LUTIN
```

```
*****************
*   GALAXIA-4   *
*****************

**************************
* SON EXPLOSION ENNEMIS *
**************************

       DATA 0
SEEN   MOV  R11,@SEEN-2
       BL   @RND
       SWPB R1
       ANDI R1,3
       A    R1,R1
       MOV  @TBLE(R1),R1
SEEN2  LI   R0,>1000
       LI   R2,38
       BLWP @VMBW
       LIMI 0
       LI   R10,>1000
       MOV  R10,@>83CC
       SOCB @UNB,@>83FD
       MOVB @UNB,@>83CE
       MOV  @SEEN-2,R11
       RT

TBLE   DATA TBLE+8,TBLE+46,TBLE+84
       DATA TBLE+84

* TABLES DE SON
* SON 1
       BYTE 5,>9F,>BF,>DF,>FF,>E3,1
       BYTE 2,>E4,>F0,20
       BYTE 2,>E4,>F1,16
       BYTE 2,>E4,>F2,13
       BYTE 2,>E4,>F4,10
       BYTE 2,>E4,>F6,10
       BYTE 2,>E4,>FA,10
       BYTE 2,>E4,>FE,10
       BYTE 1,>FF,0
* SON 2
       BYTE 5,>9F,>BF,>DF,>FF,>E3,1
       BYTE 2,>E5,>F0,20
       BYTE 2,>E5,>F1,16
       BYTE 2,>E5,>F2,13
       BYTE 2,>E5,>F4,10
       BYTE 2,>E5,>F6,10
       BYTE 2,>E5,>FA,10
       BYTE 2,>E5,>FE,10
       BYTE 1,>FF,0
* SON 3
       BYTE 5,>9F,>BF,>DF,>FF,>E3,1
       BYTE 2,>E6,>F0,20
       BYTE 2,>E6,>F1,16
       BYTE 2,>E6,>F2,13
       BYTE 2,>E6,>F4,10
       BYTE 2,>E6,>F6,10
       BYTE 2,>E6,>FA,10
       BYTE 2,>E6,>FE,10
       BYTE 1,>FF,0

***************
* ATTERRISSAGE *
***************

T1     TEXT 'ATTERRISSAGE PARFAIT'
       TEXT 'ATTERRISSAGE UN PEU DUR !'
       TEXT 'TRAIN D ATTERRISSAGE PLIE'
       TEXT 'REPARATION :'
       TEXT 'LE VAISSEAU S EST ECRASE'
REP1   TEXT ' 50 UNITES D ENERGIE'
REP2   TEXT '100 UNITES D ENERGIE'
REP3   TEXT '150 UNITES D ENERGIE'
REP4   TEXT '200 UNITES D ENERGIE'

ATTER  LI   R0,>780
       CLR  R1
       BL   @VSBR
       CI   R1,>2000
       JL   ATTER1
       CI   R1,>6000
       JL   ATTER2

*************************
*LE VAISSEAU S'EST ECRASE*
*************************

       LI   R0,68
       LI   R1,T1+82
       LI   R2,24
       BLWP @VMBW
       B    @FINJEU
```

```
*****************
*BON ATTERRISSAGE*
*****************

ATTER1 LI    R0,70
       CI    R1,>0800
       JL    ATTER3
       LI    R1,T1+20
       LI    R2,25
       A     @CINQ,@POINT
       JMP   ATTER4
ATTER3 LI    R1,T1
       LI    R2,20
       A     @DIX,@POINT
ATTER4 BLWP @VMBW
       B     @DECOL

********************
* ATTERRISSAGE RATE *
********************

ATTER5 DATA REP1,REP2,REP3,REP4
       DATA -50,-100,-150,-200
ATTER2 ANDI R1,>F000
       SRA  R1,12
       A    R1,R1
       MOV  @ATTER5(R1),R3
       MOV  @ATTER5+8(R1),R4
       LI   R0,68
       LI   R1,T1+45
       LI   R2,25
       BLWP @VMBW
       LI   R0,106
       LI   R1,T1+70
       LI   R2,12
       BLWP @VMBW
       LI   R0,135
       MOV  R3,R1
       LI   R2,20
       BLWP @VMBW
       A    R4,@ENERGI
       BL   @AFFPTS

***********
*DECOLLAGE*
***********

DEC1   TEXT 'ATTENTION AU DECOLLAGE'
D300   DATA 300
       EVEN

DECOL  BL   @DELAI
       BL   @DELAI
       BL   @DELAI
       BL   @DELAI
       BL   @DELAI
       LI   R0,>300
       LI   R1,>9A00
       BL   @VSBW
       INC  R0
       LI   R1,>4100
       BL   @VSBW
       BL   @HCHAR
       DATA 64,>2000,256
       MOV  @UN,@DRATT
       A    @D300,@ENERGI
       BL   @AFFPTS
       LI   R0,133
       LI   R1,DEC1
       LI   R2,22
       BLWP @VMBW
       BL   @DELAI
       BL   @DELAI
       BL   @DELAI
       BL   @DELAI
       BL   @SONEXP
       BL   @HCHAR
       DATA 64,32,256
       LI   R0,>780
       LI   R1,>FA00
       BL   @VSBW
       INC  R0
       CLR  R1
       BL   @VSBR
       B    @TAB25
```

```
****************
*   GALAXIA-5   *
****************

* DESSIN DU TITRE

ORI    DATA 0
ORF    DATA 0
ABI    DATA 0
ABF    DATA 0

RTITRE DATA 0
       BSS  472
TITRE  MOV  R11,@RTITRE
       LI   R0,>900
       LI   R1,RTITRE+2
       LI   R2,472
       BLWP @VMBR
       LI   R0,>1F0
       BLWP @VWTR
       LI   R0,>711
       BLWP @VWTR
       LI   R0,>1800
       CLR  R4
E1     CLR  R1
       CLR  R3
E2     BL   @VSBW
       INC  R0
       INC  R3
       AI   R1,>100
       CI   R3,>100
       JNE  E2
       INC  R4
       CI   R4,3
       JNE  E1
NE     CLR  R0
       CLR  R1
N1     BL   @VSBW
       INC  R0
       CI   R0,>1800
       JNE  N1
       LI   R0,>2000
       LI   R1,>1100
CO     BL   @VSBW
       INC  R0
       CI   R0,>3800
       JNE  CO
       LI   R1,RVDPB
TITRE8 MOV  *R1+,R0
       JLT  TITREA
       BLWP @VWTR
       JMP  TITRE8
TITREA LI   R0,>1B00
       LI   R1,>D000
       BL   @VSBW
```

```
* DESSIN DU VAISSEAU

        LI    R3,AV
        LI    R2,>5100

TITRE1  MOV   *R3+,@ABF
        MOV   @ABF,@ABF
        JEQ   TITRE2
        MOV   *R3+,@ORF
        MOV   *R3+,@ABI
        MOV   *R3+,@ORI

        BL    @LIGNE
        LI    R13,136
        S     @ABF,R13
        A     R13,R13
        A     R13,@ABF
        LI    R13,136
        S     @ABI,R13
        A     R13,R13
        A     R13,@ABI
        BL    @LIGNE
        JMP   TITRE1


* DESSIN DU TITRE



TITRE2  LI    R3,AZ
        LI    R2,>9100
TITRE3  MOV   *R3+,@ABF
        MOV   @ABF,@ABF
        JEQ   TITRE4
        MOV   *R3+,@ORF
        MOV   *R3+,@ABI
        MOV   *R3+,@ORI
        BL    @LIGNE
        JMP   TITRE3
TITRE4  BLWP  @KSCAN
        MOVB  @>837C,R0
        JEQ   TITRE4
        LI    R1,RVDPG
TITRE6  MOV   *R1+,R0
        JLT   TITRE7
        BLWP  @VWTR
        JMP   TITRE6


* REGISTRES VDP POUR MODE GRAPHIQUE


RVDPG   DATA >0000,>01E0,>0200,>030E
        DATA >0401,>0506,>0600,>0711
        DATA >FFFF


* REGISTRES VDP POUR LE BIT MAP MODE


RVDPB   DATA >0002,>01E0,>0206,>03FF
        DATA >0403,>0502,>FFFF


TTT     TEXT 'NIVEAU DE JEU'
        TEXT '1 -> DEBUTANT'
        TEXT '2 -> AMATEUR '
        TEXT '3 -> CONFIRME'
        TEXT '4 -> CHAMPION'
        TEXT '  TIRS ENNEMIS : '
        TEXT 'VITESSE ENNEMIS :'
        TEXT '  ATTERRISSAGE  :'
        TEXT 'C EST PARTI !!!'
        EVEN

TITRE7  LI    R0,>E000
        MOVB  R0,@>83D4
        LI    R0,>900
        LI    R1,RTITRE+2
        LI    R2,472
        BLWP  @VMBW
        BL    @HCHAR

        DATA  >380,>4100,18
        BL    @HCHAR
        DATA  >392,>9100,15
        BL    @EFFAC

TITRE9  LI    R0,8
        LI    R1,TTT
        LI    R2,13
        BLWP  @VMBW
        LI    R0,168
        LI    R1,TTT+13
        LI    R2,13
        BLWP  @VMBW
        AI    R0,64
        LI    R1,TTT+26
        BLWP  @VMBW
        AI    R0,64
        LI    R1,TTT+39
        BLWP  @VMBW
        AI    R0,64
        LI    R1,TTT+52
        BLWP  @VMBW
        AI    R0,126
        LI    R1,TTT+65
        LI    R2,17
        BLWP  @VMBW


* SAISIE DES NIVEAUX
        BL    @SCAN
        BL    @DELAI


*->TIR ENNEMI
        MOV   @NIVTE(R1),@TIRLIM
        LI    R1,TTT+82
        BLWP  @VMBW
        BL    @SCAN
        BL    @DELAI


*->VITESSE ENNEMIS
        MOV   @NIVVI(R1),@VITLIM
        LI    R1,TTT+99
        BLWP  @VMBW
        BL    @SCAN


*->ATTERRISAGE
        MOV   @NIVAT(R1),@VITBAR
        BL    @HCHAR
        DATA  0,>2000,768
        LI    R0,329
        LI    R1,TTT+116
        LI    R2,15
```

```
       BLWP @VMBW
       BL   @DELAI
       BL   @DELAI
       BL   @DELAI
       BL   @DELAI
       BL   @EFFAC

       MOV  @RTITRE,R11
       RT

* NIVEAUX
NIVTE  DATA 2048,>2000,>4000,>5000
NIVVI  DATA 8192,>4800,>5000,>5800
NIVAT  DATA 2048,>4000,>8000,>0000


* TESTE LES TOUCHES 1,2,3,4
       DATA 0
SCAN   MOV  R11,@SCAN-2
SCAN1  CLR  @>8374
       CLR  R1
       BLWP @KSCAN
       MOVB @>8375,R1
       CI   R1,>FF00
       JEQ  SCAN1
       SWPB R1
       AI   R1,-49
       JLT  SCAN1
       CI   R1,3
       JH   SCAN1
       A    R1,R1
       MOV  @SCAN-2,R11
       RT


* TRACE LE TRAIT
       DATA 0
LIGNE  MOV  R11,@LIGNE-2
       MOV  @ABF,R7
       MOV  @ABI,R9
       CLR  R12
       LI   R5,1
       LI   R6,1
       S    R9,R7
       JLT  LI1
       JMP  LI2
LI1    NEG  R5
       NEG  R7
LI2    MOV  R7,R7
       JNE  LI3
       SETO R12
LI3    MOV  @ORF,R8
       MOV  @ORI,R10
       S    R10,R8
       JLT  LI4
       JMP  LI5
LI4    NEG  R6
       NEG  R8
LI5    MOV  R9,R0
       MOV  R10,R1
       BL   @PT
       C    R9,@ABF
       JNE  LI6
       C    R10,@ORF
       JNE  LI6
       MOV  @LIGNE-2,R11
       RT


LI6    MOV  R12,R12
       JLT  LI7
       A    R5,R9
       S    R8,R12
       JMP  LI5
LI7    A    R6,R10
       A    R7,R12
       JMP  LI5

       DATA 0
PT     MOV  R11,@PT-2
       MOV  R4,R11
       MOV  R1,R4
       SLA  R4,5
       SOC  R1,R4
       ANDI R4,>FF07
       A    R0,R4
       ANDI R0,>0007
       S    R0,R4
       MOV  R0,R14
       MOV  R4,R0
       BL   @VSBR
       SOCB @OO(R14),R1


       BL   @VSBW
       AI   R0,>2000
       MOV  R2,R1
       BL   @VSBW
       MOV  R11,R4
       MOV  @PT-2,R11
       RT


OO     DATA >8040,>2010,>0804,>0201
AV     DATA 100,32,78,64,100,32,130,32
       DATA 86,52,108,60,108,60,117,77
       DATA 130,28,116,84,130,28,136,28
       DATA 116,84,128,84,116,84,120,96
       DATA 130,28,136,28,134,28,124,120
       DATA 124,120,133,128,120,96,127
       DATA 96
       DATA 136,119,133,119,133,119,133
       DATA 132
       DATA 133,132,136,132,134,132,134
       DATA 145
       DATA 128,84,136,88,136,88,136,119
       DATA 136,52,136,0,132,68,128,84
       DATA 120,4,136,4,120,4,116,9
       DATA 116,9,136,13,134,52,136,52
       DATA 132,68,136,68,132,68,134,52
       DATA 0


AZ     DATA 56,164,52,156,52,156,28,156
       DATA 28,156,47,188,47,188,67,188
       DATA 67,188,59,172,59,172,52,172
       DATA 72,188,72,156,72,156,93,188
       DATA 72,172,83,172
       DATA 92,156,100,188,100,188,120
       DATA 188
       DATA 125,188,136,156,136,156,147
       DATA 188
       DATA 130,172,141,172
       DATA 152,188,180,156,156,156,172
       DATA 188
       DATA 184,188,192,188,196,156,204
       DATA 156
       DATA 188,188,200,156
       DATA 204,188,232,156,232,156,224,188
       DATA 188,218,172,228,172,0
```

Logo et la récursivité

Michel Schurmans

Le premier coup d'euil sur le Logo TI m'avait donné l'impression d'un langage simple, pour ne pas dire simpliste. En fait, il s'agit d'un langage à la fois plus souple, plus concis et plus précis que le Basic.
En simplifiant à l'extrême, les trois notions fondamentales du Logo sont les listes, les procédures et la récursivité.

La liste

Il s'agit d'une collection d'éléments, ou "items". Chacun de ces éléments peut être lui-même une collection d'éléments, et ainsi de suite...

La liste simple s'écrira ainsi :
[POMME POIRE SCOUBIDOU]
et ce sans oublier un espace avant le crochet "]".

Une liste composée pourra s'écrire :
[[POMME POIRE SCOUBIDOU] POISSON [TABLE CHAISE]]

Cette dernière liste comporte trois éléments et surtout pas six. Une liste peut ne pas avoir d'élément; on l'appelle une liste vide.

La procédure

C'est une collection d'instructions. Le mot instruction ayant le même sens qu'en Basic, si la procédure est une collection d'instructions, on peut dire qu'une procédure est une liste d'instructions exécutables par le TI-99.

Pour illustrer ceci, nous allons étudier le moyen de mémoriser un dessin fait à l'aide de la tortue.
Pour dessiner un carré, on peut écrire :

```
AVANCE 40 <ENTER>
DROITE 90 <ENTER>
AV 40 <ENTER>
DR 90 <ENTER>
AV 40 DR 90 AV 40 DR 90 <ENTER>
```

Si nous voulons redessiner le carré, il faut recommencer la même séquence d'instructions. Pour un carré, ce n'est pas grave; par contre, si vous avez dessiné une maison avec ses fenêtres, sa porte, etc... c'est beaucoup plus long.

Nous pouvons envisager d'enregistrer les ordres donnés à la tortue dans une liste, puis de réexécuter cette liste comme il nous plait. De plus, comme nous dessinons "en direct", nous pouvons gommer nos erreurs; nos corrections seront enregistrées.

La procédure MEMORISATION permettra cet enregistrement. Les numéros entre parenthèses renvoient aux commentaires : ne pas les taper au clavier.

```
(1)  POUR MEMORISATION
(2)  CREE "INSTRUCTION LISLIGNE
(3)  SI :INSTRUCTION = [ ] ALORS STOP
(4)  EXECUTE :INSTRUCTION
(5)  CREE "SUITEINSTR PHRASE :SUITEINSTR :INSTRUCTION
(6)  MEMORISATION
(7)  FIN
```

1) MEMORISATION est le nom de la procédure créée;

2) Dès que vous appuierez sur <ENTER>, la ou les instructions tapées seront mémorisées dans une liste qui porte le nom "INSTRUCTION";

3) :INSTRUCTION signifie : les instructions contenues dans la variable (liste) INSTRUCTION. En Logo, on distingue le contenant du contenu. Soit une liste dénommée INSTRUCTION. Si on fait appel à la notion de contenant, on l'écrira :
"INSTRUCTION
et par le contenu :
:INSTRUCTION

Ceci est une notion fondamentale de l'informatique, malheureusement implicite dans la plupart des langages.
Le but de cette troisième ligne est de tester si nous avons entré quelque chose avant <ENTER>. Si oui, la liste INSTRUCTION n'est pas vide et la procédure passe à la quatrième ligne. Si, au contraire, vous avez fait <ENTER> directement, la procédure va s'arrêter.

4) Les instructions entrées dans "INSTRUCTION sont maintenant exécutées.

5) Nous créons une nouvelle liste SUITEINSTR. Le contenant "SUITEINSTR reprend le contenu de :SUITEINSTR et y ajoute le contenu :INSTRUCTION.

6) On continue en appelant à nouveau la présente procédure. A noter : MEMORISATION n'est précédé d'aucun signe tel que " ou :.

Pour exécuter la suite d'instructions ainsi entrée, il faut écrire EXECUTE :SUITEINSTR; c'est-à-dire donner l'ordre au TI d'exécuter les instructions contenues dans la liste "SUITEINSTR. Ceci est assez long à écrire, mais le Logo TI permet d'y remédier en écrivant une deuxième procédure :

```
POUR EX
EXECUTE :SUITEINSTR
FIN
```

Il suffira de taper EX pour voir s'exécuter le dessin.

Si vous voulez compléter votre dessin, il suffira de relancer la procédure MEMORISATION. Par contre, si vous voulez faire un autre dessin, il faut penser à vider la liste "SUITEINSTR. Plutôt que de le faire à chaque fois, écrivons une troisième procédure en y ajoutant d'autres instructions qu'il faudrait répéter.

```
(8)  POUR DEBUT
(9)  CREE "SUITEINSTR [ ]
(10) APPELLE TORTUE
(11) EFFACEECRAN
(12) MEMORISATION
(13) FIN
```

Cette troisième procédure introduit deux notions importantes :

- En ligne 9, nous avons demandé l'exécution d'une simple instruction; en 12 nous avons demandé l'exécution, d'une procédure, c'est-à-dire, en fait, de sept instructions. D'où la notion : une suite d'instructions peut être considérée comme une instruction simple. La réciproque est vraie et se trouve à la base de la programmation structurée (ou descendante).
- Puisque nous n'avons pas le courage d'écrire à chaque fois les lignes 9, 11 et 12, nous avons écrit une fois pour toutes une procédure. "DEBUT" est plus rapide à saisir que les lignes qui suivent. Par contre, pour que la procédure marche même avant d'être dans le monde de la tortue, nous avons ajouté la ligne 10.

 Cette ligne ne sert que la première fois, mais il n'est pas grave de l'exécuter inutilement.

 Ceci illustre un autre principe important : soyons fainéants et laissons l'ordinateur travailler !

Maintenant, après avoir souffert en entrant ces trois procédures, exécutons-les en dessinant notre carré. Nous aurons à taper :

DEBUT <ENTER>
AV 40 <ENTER>
DR 90 <ENTER>
AV 40 <ENTER>
DR 90 <ENTER>
AV 40 DR 90 AV 40 DR 90 <ENTER>
<ENTER> (pour dire que le dessin est terminé)
EE (pour effacer l'écran)
EX (pour redessiner le carré)

Si nous tapons AFFICHE :SUITE INSTR, le TI nous donne :

[AV 40 DR 90 AV 40 DR 90 [AV 40 DR 90 AV 40 DR 90]]

Ceci correspond à la façon dont nous avons entré nos ordres de dessin.

Si vous entrez un dessin plus complexe que notre malheureux carré, vous pouvez obtenir le message "Plus d'espace disponible au niveau X" (60 par exemple). L'origine de cette plaisanterie est à la ligne 6; quand nous appelons **à nouveau** la procédure MEMORISATION, nous n'exécutons pas la même procédure :

- la première fois nous exécutons la procédure numéro 1;
- la deuxième fois nous exécutons la procédure numéro 2;
- la soixantième fois nous exécutons la procédure numéro 60.

Pour le TI-99, il y a 60 procédures différentes donc 60 fois plus d'espace mémoire utilisé. Ceci n'est pas une carence du Logo, ni même du TI; nous retrouvons cette contrainte dans tous les langages récursifs et sur tous les systèmes informatiques. Ce n'est que la contrepartie de la puissance des procédures récursives, comme nous vous l'expliquerons dans un prochain article.

□

Dessinez avec le Basic Etendu

Georges Herbepin

Ce programme permet de réaliser des dessins directement à l'écran, en partant de motifs prédéfinis.

Compte tenu de la place mémoire disponible sur la console de base, l'espace "feuille de papier" a été réduit à 24 lignes par 25 colonnes (quatre colonnes de motifs prédéfinis plus 3 colonnes supprimées). La même raison nous a empêché d'écrire le mode d'emploi dans le programme.

Il existe dans cette version 96 motifs prédéterminés, apparaissant dans les quatre colonnes de gauche. Il est possible d'en ajouter, mais cela obligerait à refaire les calculs. Vous pouvez aussi redéfinir les motifs à votre convenance, pour réaliser un autre type de dessin. A tout moment, vous pouvez enregistrer votre dessin sur cassette pour le reprendre plus tard, ou tout simplement le visualiser.

Mode d'emploi

- La touche Alpha Lock doit être en position basse.
- Pour déplacer le "crayon" (recherche et déplacement du motif), il faut d'abord faire *1 et ensuite utiliser les touches D, E, S, X, W, R, C ET Z.
- Pour augmenter la vitesse de déplacement du crayon sur l'écran, il faut presser la touche "+" (au début du programme, la vitesse rapide est obtenue par défaut).
- Pour diminuer la vitesse de déplacement : "-".
- L'opération en cours peut être annulée à tout moment grâce à la touche ".".
- Prise (première pression) et inscription du motif (seconde pression) par la touche "espace".
- Après inscription du motif (seconde pression sur "espace"), choix du mode d'affichage par les touches "H" pour inscription horizontale, "V" pour inscription verticale et "1" pour un seul affichage.
- Il y a trois niveaux de choix du nombre de répétitions de 0 à 999, limités aux cases restantes : aux deux premiers niveaux, possibilité d'inscrire le nombre de répétitions déjà choisi par la touche "espace". Au troisième niveau, affichage automatique du total.
- Enregistrement sur cassette (CS1) en cours de dessin, par la touche "FCTN-1".
- Au début du programme, possibilité de transfert en mémoire d'un dessin depuis une cassette, pour visualisation ultérieure.
- A tout moment, visualisation du dessin en mémoire par "FCTN 9".

Il faut environ huit minutes pour enregistrer ou lire un dessin vers ou depuis une cassette.

```
100 ! ************************
110 ! *                      *
120 ! * Programme de dessin  *
130 ! *                      *
140 ! ************************
150 ! *                      *
160 ! *     Basic Etendu     *
170 ! *                      *
180 ! ************************
190 !
200 CALL CLEAR :: DISPLAY AT(4,10):"* DES
SIN *" :: DISPLAY AT(7,1):"Entrer un de
ssin ?(O/N)" :: OPTION BASE 1 :: DIM W(24
,25)
210 CALL KEY(0,K,S):: IF K=78 THEN 290
220 IF K=79 THEN 240
230 GOTO 210
240 OPEN #1:"CS1",INTERNAL,INPUT ,FIXED 1
28
250 FOR I=1 TO 24
260 INPUT #1:W(I,1),W(I,2),W(I,3),W(I,4),
W(I,5),W(I,6),W(I,7),W(I,8),W(I,9),W(I,
10),W(I,11),W(I,12),W(I,13)
270 INPUT #1:W(I,14),W(I,15),W(I,16),W(I,
17),W(I,18),W(I,19),W(I,20),W(I,21),W(I
,22),W(I,23),W(I,24),W(I,25)
280 NEXT I :: CLOSE #1
290 CALL CLEAR :: CALL CHAR(37,"FFFFFFFFF
FFFFFFF")
300 CALL CHAR(33,"0000081C3E1C0800183C3C3
C18001800004428101000000000006314080814
00"):: CALL COLOR(1,2,8):: CALL CHAR(37,"
FFFFFFFFFFFFFFFF")
310 CALL CHAR(40,"FFFFFFFFFFFFFFFF0103070
F1F3F7FFF80C0E0F0F8FCFEFFFF7F3F1F0F0703
01")
320 CALL CHAR(44,"FFFEFCF8F0E0C080223C2E6
8783000000000006314080000000000000000
00"):: CALL COLOR(2,16,8)
330 CALL CHAR(48,"FFFFFFFFFFFFFFFF0103070
F1F3F7FFF80C0E0F0F8FCFEFF88CCEAF9F8FCFE
FF")
340 CALL CHAR(52,"103854BA549238100143E74
FDF7F7FFF80C2E7F2FBFEFEFF0000000080C0A0
90"):: CALL COLOR(3,4,8)
350 CALL CHAR(56,"FFFFFFFFFFFFFFFF0103070
F1F3F7FFF80C0E0F0F8FCFEFF081C3E3E1C0808
08")
360 CALL CHAR(60,"FF7F3F1F0F0703011113E7A
F3F3F7FFF82C9E5F2FAFEFEFFFFFEFCF8F0E0C0
80"):: CALL COLOR(4,13,4)
370 CALL CHAR(64,"FFFFFFFFFFFFFFFF0103070
F1F3F7FFF80C0E0F0F8FCFEFFFFFEFCF8F0E0C0
80")
380 CALL CHAR(68,"FF7F3F1F0F0703010000140
81400000000000000010281008081C14367EFF
FF"):: CALL COLOR(5,7,4)
390 CALL CHAR(72,"FFFFFFFFFFFFFFFF0103070
F1F3F7FFF80C0E0F0F8FCFEFFFFFEFCF8F0E0C0
80")
400 CALL CHAR(76,"FF7F3F1F0F070301FFFFE3C
78FC1E7FFF77BBDDBD4D3E7F7FFFFC3A581BD7E
FF"):: CALL COLOR(6,5,4)
410 CALL CHAR(80,"FFFFFFFFFFFFFFFFFFC1D5C
1D5C1FFFFFFE7CBADADA5ADADFFFF005A00FFFF
FF")
420 CALL CHAR(84,"FFFFC3C3C3C7C3C3FFF7E3F
7FFFFFFFF"):: CALL COLOR(7,12,14)
430 CALL CHAR(88,"0203161AB0D080808080D0B
01A160302F8FCFC9E16122201000000009292FF
92")
440 CALL CHAR(92,"92FF9292000000000000083
050F81C1F0F07050C1424222010000000000003
F5"):: CALL COLOR(8,14,4)
450 CALL CHAR(96,"FFFFFFFFFFFFFFFF0103070
F1F3F7FFFFF7F3F1F0F07030180C0E0F0F8FCFE
FF")
460 CALL CHAR(100,"FFFEFCF8F0E0C080000008
183C2E000000060E13E3E529502030C5C3B102
828"):: CALL COLOR(9,15,4)
470 CALL CHAR(104,"0028102802450200002214
18101008080000001048A81410")
480 CALL CHAR(107,"4122362A1C2A220000018
3C3C7E7E7E9249490024929200"):: CALL COL
OR(10,12,13)
490 CALL CHAR(112,"0818483C1A182844101812
3C5818142200000042253C1824AAFFAAFFAAFFA
AFF"):: CALL COLOR(11,10,15)
500 CALL CHAR(120,"082A2A6B08FF7E00000002
83447C3800810081BDDBE7FFE7FBF9C773B5CFE
FF3")
510 CALL CHAR(124,"00000001030B2B2A286B6B
EB08FF553F002028A8A8ACAC2CAEAEAF2F20FF5
7FE"):: CALL COLOR(12,16,5)
520 CALL CHAR(128,"01014121110B0307FF0303
0508102000000000020488D0C0FFE0C0D08884828
080")
530 CALL CHAR(132,"0000631408080814FFFF0F
2F0FFFFFFFFFFFF0F4F0FFFFFF"):: CALL COL
OR(13,12,8)
540 CALL CHAR(136,"0103070F1F3F7FFF80C0E0
F0F8FCFEFFFFFEFCF8F0E0C080FF7F3F1F0F070
301")
550 CALL CHAR(140,"FFFF9F1EC1C1AD6AFFFFF9
788383B556"):: CALL COLOR(14,13,15):: C
ALL SCREEN(2)
560 CALL VCHAR(1,1,37,48):: CALL VCHAR(1,
32,37,24):: B=33 :: FOR A=1 TO 3 :: B=B
+1 :: CALL VCHAR(A,3,B):: NEXT A
570 B=39 :: FOR A=4 TO 24 :: B=B+1 :: CAL
L VCHAR(A,3,B):: NEXT A
580 B=60 :: FOR A=1 TO 24 :: B=B+1 :: CAL
L VCHAR(A,4,B):: NEXT A
590 CALL VCHAR(1,5,85):: B=87 :: FOR A=2
TO 23 :: B=B+1 :: CALL VCHAR(A,5,B):: N
EXT A
600 CALL VCHAR(24,5,112):: B=112 :: FOR A
=1 TO 3 :: B=B+1 :: CALL VCHAR(A,6,B)::
 NEXT A
610 B=119 :: FOR A=4 TO 18 :: B=B+1 :: CA
LL VCHAR(A,6,B):: NEXT A
620 B=135 :: FOR A=19 TO 24 :: B=B+1 :: C
ALL VCHAR(A,6,B):: NEXT A
630 CALL SPRITE(#1,33,2,90,70):: P,R=0
640 CALL KEY(0,K,S)
650 IF K=3 THEN 1540
660 IF K=15 THEN GOSUB 1610
670 IF K=61 THEN Z=0 :: GOTO 690
680 IF K=47 THEN Z=1
690 IF K=68 THEN X=0 :: Y=8 :: GOTO 780
700 IF K=69 THEN X=-8 :: Y=0 :: GOTO 780
710 IF K=83 THEN X=0 :: Y=-8 :: GOTO 780
720 IF K=88 THEN X=8 :: Y=0 :: GOTO 780
730 IF K=87 THEN X=-8 :: Y=-8 :: GOTO 780
740 IF K=82 THEN X=-8 :: Y=8 :: GOTO 780
750 IF K=67 THEN X=8 :: Y=8 :: GOTO 780
760 IF K=90 THEN X=8 :: Y=-8 :: GOTO 780
770 IF S=0 THEN X=0 :: Y=0
780 CALL POSITION(#1,A,B)
790 IF A<16 OR A>171 THEN 810
800 IF B<52 OR B>228 THEN 810 ELSE 820
810 IF Z=0 THEN Z=1
820 IF Z=1 THEN X=X/4 :: Y=Y/4
830 IF A<4 THEN CALL LOCATE(#1,4,B)
```

```
840 IF A>182 THEN CALL LOCATE(#1,182,B)
850 IF B<18 THEN CALL LOCATE(#1,A,18)
860 IF B>239 THEN CALL LOCATE(#1,A,239)
870 CALL MOTION(#1,X,Y)
880 IF S=-1 THEN 910
890 IF K=32 THEN GOSUB 930
900 IF K=46 THEN 1070
910 IF P=1 THEN 1020
920 GOTO 640
930 CALL MOTION(#1,0,0)
940 IF P=1 THEN 1040
950 P=1 :: A1=INT(A/8):: B1=INT(B/8):: A2
=A1*8 :: B2=B1*8 :: A3=A-A2 :: B3=B-B2
960 IF A3<=4 THEN A4=A1+1 :: A5=A1*8 :: G
OTO 980
970 A4=A1+2 :: A5=(A1+1)*8
980 IF B3<=4 THEN B4=B1+1 :: B5=B1*8 :: G
OTO 1000
990 B4=B1+2 :: B5=(B1+1)*8
1000 IF A5=0 THEN A5=4
1010 CALL LOCATE(#1,A5,B5):: CALL GCHAR(A
4,B4,M):: RETURN
1020 IF R=0 THEN CALL SOUND(20,880,0):: C
ALL HCHAR(A4,B4,32):: R=1 :: GOTO 640
1030 IF R=1 THEN CALL SOUND(20,660,0):: C
ALL HCHAR(A4,B4,M):: R=0 :: GOTO 640
1040 C1=INT(A/8):: D1=INT(B/8):: C2=C1*8
:: D2=D1*8 :: C3=A-C2 :: D3=B-D2
1050 IF D1<5 THEN 1070
1060 IF D1=5 THEN 1080 ELSE 1090
1070 CALL SOUND(30,110,5,220,10,330,15)::
 P,R=0 :: CALL HCHAR(A4,B4,M):: GOTO 64
0
1080 IF D3<=4 THEN 1070
1090 IF C3<=4 THEN C4=C1+1 :: C5=C1*8 ::
GOTO 1110
1100 C4=C1+2 :: C5=(C1+1)*8
1110 IF D3<=4 THEN D4=D1+1 :: D5=D1*8 ::
GOTO 1130
1120 D4=D1+2 :: D5=(D1+1)*8
1130 IF C5=0 THEN C5=1
1140 P,R=0 :: CALL LOCATE(#1,C5,D5):: CAL
L HCHAR(A4,B4,M)
1150 CALL KEY(0,K,S):: IF S=0 THEN 1200
1160 IF K=72 THEN F=0 :: GOTO 1220
1170 IF K=86 THEN F=1 :: GOTO 1220
1180 IF K=49 THEN 1210
1190 IF K=46 THEN 1070
1200 CALL SOUND(20,440,10):: GOTO 1150
1210 CALL HCHAR(C4,D4,M):: GOTO 640
1220 CALL KEY(0,K,S):: IF S=0 THEN 1250
1230 IF K=46 THEN 1070
1240 IF K>=48 AND K<=57 THEN 1260
1250 CALL SOUND(20,330,10):: GOTO 1220
1260 FOR T=1 TO 50 :: NEXT T :: K=K-48 ::
 K1=K
1270 CALL KEY(0,K,S):: IF S=0 THEN 1310
1280 IF K=46 THEN 1070
1290 IF K=32 THEN K3=K1 :: GOTO 1380
1300 IF K>=48 AND K<=57 THEN 1320
1310 CALL SOUND(20,220,10):: GOTO 1270
1320 FOR T=1 TO 50 :: NEXT T :: K=K-48 ::
 K2=K
1330 CALL KEY(0,K,S):: IF S=0 THEN 1370
1340 IF K=46 THEN 1070
1350 IF K=32 THEN K3=K2+(K1*10):: GOTO 13
80
1360 IF K>=48 AND K<=57 THEN K=K-48 :: K3
=K+(K2*10)+(K1*100):: GOTO 1380
1370 CALL SOUND(20,110,10):: GOTO 1330
1380 IF K3=0 THEN 640
1390 IF K3=1 THEN 1210
1400 IF F=0 THEN 1430 ELSE C6=((31-D4)*24
)+(24-(C4-1))
1410 IF K3>C6 THEN K3=C6
1420 CALL VCHAR(C4,D4,M,K3):: GOTO 640
1430 K4=31-(D4-1):: C6=((24-C4)*25)+K4
1440 IF K3>C6 THEN K3=C6
1450 IF K3<=K4 THEN CALL HCHAR(C4,D4,M,K3
):: GOTO 640
1460 K5=K3-K4 :: K6=INT(K5/25):: K7=K3-((
K6*25)+K4):: CALL HCHAR(C4,D4,M,K4)
1470 IF K6=0 THEN 1510
1480 IF K6=1 THEN 1520
1490 FOR C7=C4+1 TO C4+K6 :: CALL HCHAR(C
7,7,M,25):: NEXT C7
1500 IF K7=0 THEN 640 ELSE CALL HCHAR(C7,
7,M,K7):: GOTO 640
1510 IF K7=0 THEN 640 ELSE CALL HCHAR(C4+
1,7,M,K7):: GOTO 640
1520 CALL HCHAR(C4+1,7,M,25)
1530 IF K7=0 THEN 640 ELSE CALL HCHAR(C4+
2,7,M,K7):: GOTO 640
1540 CALL PATTERN(#1,32):: FOR I=1 TO 24
:: FOR J=7 TO 31 :: CALL GCHAR(I,J,M)::
 K=J-6 :: W(I,K)=M :: NEXT J :: NEXT I
1550 CALL CLEAR :: FOR A=1 TO 12 :: CALL
COLOR(A,2,8):: NEXT A :: CALL CHARSET :
: CALL SCREEN(8):: CALL CLEAR
1560 OPEN #1:"CS1",INTERNAL,OUTPUT,FIXED
128
1570 FOR I=1 TO 24
1580 PRINT #1:W(I,1),W(I,2),W(I,3),W(I,4)
,W(I,5),W(I,6),W(I,7),W(I,8),W(I,9),W(I
,10),W(I,11),W(I,12),W(I,13)
1590 PRINT #1:W(I,14),W(I,15),W(I,16),W(I
,17),W(I,18),W(I,19),W(I,20),W(I,21),W(
I,22),W(I,23),W(I,24),W(I,25)
1600 NEXT I :: CLOSE #1 :: END
1610 FOR I=1 TO 24 :: FOR J=1 TO 25 :: E=
W(I,J):: K=J+6 :: CALL HCHAR(I,K,E):: N
EXT J :: NEXT I :: RETURN
```



Accès au mode Bit-Map en Basic

Gerard Baroni et Jean Marin

NDLR : le système ici employé pour accéder au mode Bit-Map laisse très peu de place pour le programme Basic et n'est donc pas utilisable avec un contrôleur de disquettes, puisque celui-ci utilise la partie haute de la mémoire vidéo.

Le Bit-Map est le mode graphique le plus puissant de notre TI-99/4A. Il permet d'allumer ou d'éteindre les 49152 pixels qui constituent notre écran de télévision.

La compatibilité de ce mode graphique haute résolution et du Basic, dont la facilité d'emploi n'est plus à démontrer, est désormais possible pour tout possesseur de Mini-mémoire.

Dans le numéro 6 de "99 Magazine", nous vous proposions un article sur l'écriture directe dans les registres VDP, à l'aide d'un CALL PEEKV(A,X), A étant calculé à l'aide de notre formule

*A=-32768+256*N+D.*

Cette formule nous a permis de mettre au point, en Basic et avec la Mini-mémoire, l'accès au mode Bit-Map. L'accès à ce mode est un peu lent car il faut initialiser tous les registres du processeur vidéo de la façon suivante :

R0 = 2 : mode Bit-Map *(CALL PEEKV(-32766,X));*
R1 : inchangé;
R2 = 6 : met la table d'écran de 6144 à 6911;
R3 = 255 : met la table des couleurs de 9192 à 14335. Elle définit des couleurs sur 6Ko au lieu de 16 octets, ce qui constitue tout de même une énorme différence;
R4 = 3 : met la table des formes de 0 à 6143;
R5 = 54 : met la table de description des lutins de 6912 à 7039;
R6 = 3 : met la table des formes des lutins de 6144 à 9191. La zone 6144 à 6911 est déjà utilisée par l'écran, il reste donc 128 formes possibles pour les lutins;
R7 : contient simplement la couleur d'écran et un *CALL SCREEN* suffira.

Les huit registres étant positionnés, il faut par ailleurs empêcher le Basic de se mélanger avec le Bit-Map. Il suffira pour cela de déplacer la pile, empêchant ainsi les variables de monter et la pile de se mélanger. Il faudra mettre cette pile en 14336, ce qui permet d'avoir exactement deux Ko disponibles pour le programme Basic (jusqu'en 16383). CALL LOAD(-31890, 56,0) réalise ce déplacement.

Le mode Bit-Map, à une restriction près, peut maintenant fonctionner. Quel est cet "à peu près" ? Entre 958 et 1007, le Basic occupe encore des octets et, pour l'instant, 6 carreaux de la quatrième ligne sont perdus. La solution adoptée pour résoudre ce problème consiste à échanger ces 6 carreaux contre 6 carreaux invisibles de la colonne numéro 1 qui, comme chacun le sait, n'apparaît pas sur nos téléviseurs français, sauf avec un réglage particulier.
Sans entrer dans le détail du programme dont le listing est joint, nous précisons cependant quelques points :

- la numérotation des cases est faite a priori. Comme on ne peut pas numéroter de 0 à 767, on numérote de 0 à 255 trois fois de suite;
- la table des formes fonctionne exactement comme en Basic standard. Il y a 768 formes de 8 octets chacune, soit 6144 octets. La forme contient des points allumés et des points éteints comme avec *CALL CHAR,* mais nous avons trois tables de 256 caractères chacune. Le carreau comporte huit lignes de huit pixels qui peuvent être allumés ou éteints;
- la table des couleurs : à chaque carreau correspond huit octets; il y a donc 768 fois huit octets. A chaque segment de huit pixels d'un carreau correspond un octet dans la table des couleurs. Cet octet va contenir deux couleurs de 0 à 15 (la couleur des points allumés et celle des points éteints). Au lieu d'attribuer une couleur à huit formes, on l'attribue à 1/8ème de forme. On met donc dans cet octet 16 fois la couleur des points allumés plus celle des points éteints;
- l'initialisation du programme (ligne 100 à 350) est suivie d'un calcul traçant une courbe. Les lignes 360 à 500 permettent d'allumer des pixels;
- vous pourrez appliquer vos propres formules pour calculer Y, X et C pour la couleur. Nous effectuons des calculs binaires programmés en Basic pour aller lire l'ancienne forme et la modifier. Le pixel est ensuite allumé et colorié;
- le programme se termine par un classique test du clavier; une intervention sur celui-ci restaure le mode standard en remettant 0 dans R0.

A l'exécution de ce programme, vous constaterez que nous arrivons un peu aux limites du Basic pour ce qui est de la vitesse d'exécution.

En langage machine, l'initialisation par un *CALL LINK("INIT"),* puis l'emploi des routines *"POINT", "DROITE",* etc... nous procure une vitesse beaucoup plus agréable mais, ceci est une autre histoire que nous vous conterons une prochaine fois.

□

```
100 REM  ********************
110 REM  *  Le mode  BitMap  *
120 REM  *   avec le Basic   *
130 REM  ********************
140 REM  *    Basic TI  +    *
150 REM  *   Mini  memoire   *
160 REM  ********************
170 REM
```

```
180 REM  INIT
190 CALL SCREEN(1)
200 CALL LOAD(-31890,56,0)
210 CALL CLEAR
220 FOR I=0 TO 8
230 P(I)=2^I
240 NEXT I
250 CALL PEEKV(-32766,X,"",-32250,X,"",-3
1745,X,"",-31741,X,"",-31434,X,"",-3122
9,X)
260 CALL POKEV(6912,208)
270 FOR I=0 TO 2
280 FOR A=0 TO 255 STEP 8
290 CALL POKEV(A+256*I+6144,A,A+1,A+2,A+3
,A+4,A+5,A+6,A+7)
300 NEXT A
310 NEXT I
320 FOR I=1 TO 7
330 CALL POKEV(6144+32*I,0)
340 NEXT I
350 CALL POKEV(6263,32,64,96,128,160,192,
224)
360 CALL SOUND(-100,440,0)
370 FOR A=0 TO 951 STEP 8
380 CALL POKEV(A,0,0,0,0,0,0,0,0)
390 NEXT A
400 FOR A=1008 TO 6143 STEP 8
410 CALL POKEV(A,0,0,0,0,0,0,0,0)
420 NEXT A
430 CALL SOUND(-100,440,0)
440 REM  COURBE
450 C=32
460 FOR T=0 TO 360
470 Y=96+INT(90*COS(T/56))
480 X=128+INT(110*SIN(T/14))
490 V=INT(Y/8)
500 H=INT(X/8)
510 N=7-X+8*H
520 A=248*V+8*H+Y
530 A=A+(A>951)*(A<1008)*(248*H+7936*V-30
208)
540 D1=P(N)
550 CALL PEEKV(A,D)
560 D1=D+D1+D1*(D-P(N+1)*INT(D/P(N+1)))>=P
(N))
570 CALL POKEV(A,D1,"",8192+A,C)
580 NEXT I
590 REM  FIN
600 CALL KEY(5,R,E)
610 IF E=0 THEN 600
620 CALL PEEKV(-32768,X)
```

Music Box Dancer

Jacques Mercier

Comme son nom l'indique, "Music Box Dancer" est un programme musical, écrit en **Basic TI**.

Vous pouvez choisir différents paramètres, comme la durée des notes, le tempo, l'accompagnement, etc...

Le mode d'emploi étant de le programme, nous vous laissons exécuter ce programme qui prouve, si besoin était, que notre TI est aussi un excellent musicien.

□

```
100 REM  *********************
110 REM  * Music Box Dancer  *
120 REM  *********************
130 REM  *      Basic TI     *
140 REM  *********************
150 REM
160 CALL CLEAR
170 REM .....................
180 REM
190 REM        NOTES
200 REM
210 REM .....................
220 A=131
230 B=147
240 C=165
250 D=175
260 E=196
270 G=262
280 H=294
290 I=330
300 J=349
310 K=392
320 L=440
330 M=523
340 N=587
350 O=659
360 P=698
370 Q=784
380 R=880
```



```
390 S=1047
400 REM .....................
410 REM
420 REM  AFFICHAGE DU TITRE
430 REM
440 REM .....................
450 PRINT "    *********************"
460 PRINT "    *  MUSIC BOX DANCER  *"
470 PRINT "    *********************"
480 FOR F=1 TO 8
490 PRINT
500 NEXT F
510 CALL SOUND(1000,131,0,392,2,330,1)
520 FOR DEL=1 TO 900
530 NEXT DEL
540 FOR F=1 TO 30
550 PRINT
560 NEXT F
570 PRINT "VOULEZ VOUS CHOISIR LE        RY
THME ET LES VOLUMES?(O/N)"
580 FOR F=1 TO 9
590 PRINT
600 NEXT F
610 CALL SOUND(120,392,0)
620 CALL KEY(0,KY,ST)
630 IF ST=0 THEN 620
640 CALL SOUND(250,1047,0,392,0,131,0)
650 IF KY=79 THEN 730
660 IF KY=78 THEN 670 ELSE 620
```

```
670 X=275
680 Z=2*X
690 T=0
700 Y=8
710 W=4
720 GOTO 3060
730 REM ....................
740 REM
750 REM        CHOIX DES
PARAMETRES
760 REM
770 REM ....................
780 FOR F=1 TO 15
790 PRINT
800 NEXT F
810 PRINT "              TEMPO?    "
820 PRINT "              ######"
830 PRINT
840 PRINT "           APPUYEZ SUR:"
860 PRINT " 1 POUR 300 MILLISECONDES"
870 PRINT
880 PRINT " 2 POUR 400 MILLISECONDES"
890 PRINT
900 PRINT " 3 POUR 450 MILLISECONDES"
910 PRINT
920 PRINT " 4 POUR 500 MILLISECONDES"
930 PRINT
940 PRINT " 5 POUR 550 MILLISECONDES"
950 PRINT
960 PRINT " 6 POUR 600 MILLISECONDES"
970 CALL SOUND(120,392,0)
980 CALL KEY(0,KEY,ST)
990 IF ST=0 THEN 980
1000 CALL SOUND(350,131,0,392,0,1047,0)
1010 IF KEY=49 THEN 1070
1020 IF KEY=50 THEN 1100
1030 IF KEY=51 THEN 1130
1040 IF KEY=52 THEN 1160
1050 IF KEY=53 THEN 1190
1060 IF KEY=54 THEN 1220 ELSE 980
1070 X=150
1080 Z=300
1090 GOTO 1250
1100 X=200
1110 Z=400
1120 GOTO 1250
1130 X=225
1140 Z=450
1150 GOTO 1250
1160 X=250
1170 Z=500
1180 GOTO 1250
1190 X=275
1200 Z=2*X
1210 GOTO 1250
1220 X=300
1230 Z=600
1240 GOTO 1250
1250 CALL CLEAR
1260 REM ...................
1270 REM
1280 REM     CHOIX DES
 VOLUMES
1290 REM
1300 REM ...................
1310 PRINT "    ATTENTION LE VOLUME
    DECROIT DE 0 A 9 !...      -------
 "
1320 FOR F=1 TO 8
1330 PRINT
1340 NEXT F
1350 FOR DEL=1 TO 1500
1360 NEXT DEL
1370 CALL CLEAR
1380 PRINT "          CHOISISSEZ LE
     VOLUME DU CHANT"
1390 PRINT
1400 PRINT
1410 PRINT
1420 PRINT "            (0 A 9)"
1430 PRINT
1440 PRINT "(S POUR SUPPRIMER LE CHANT)"
1450 PRINT
1460 PRINT
1470 CALL SOUND(120,1047,0)
1480 CALL KEY(0,KEY,ST)
1490 IF ST=0 THEN 1480
1500 CALL SOUND(350,131,0,392,0,1047,0)
1510 IF KEY=57 THEN 1620
1520 IF KEY=56 THEN 1640
1530 IF KEY=55 THEN 1660
1540 IF KEY=54 THEN 1680
1550 IF KEY=53 THEN 1700
1560 IF KEY=52 THEN 1720
1570 IF KEY=51 THEN 1740
1580 IF KEY=50 THEN 1760
1590 IF KEY=49 THEN 1780
1600 IF KEY=48 THEN 1800
1610 IF KEY=83 THEN 1820 ELSE 1480
1620 T=9
1630 GOTO 1840
1640 T=8
1650 GOTO 1840
1660 T=7
1670 GOTO 1840
1680 T=6
1690 GOTO 1840
1700 T=5
1710 GOTO 1840
1720 T=4
1730 GOTO 1840
1740 T=3
1750 GOTO 1840
1760 T=2
1770 GOTO 1840
1780 T=1
1790 GOTO 1840
1800 T=0
1810 GOTO 1840
1820 T=30
1830 GOTO 1840
1840 REM  ...................
1850 REM
1860 REM      CHOIX DU
 VOLUME DE                L'ACCOMPAGNEM
ENT
1870 REM
1880 REM ...................
1890 CALL CLEAR
1900 PRINT "          CHOISISSEZ LE
 VOLUME DE L'ACCOMPAGNEMENT"
1910 PRINT
1920 PRINT
1930 PRINT
1940 PRINT
1950 PRINT
1960 PRINT "           (0 A 9)"
1970 PRINT
1980 PRINT "   (S POUR LE SUPPRIMER)"
1990 PRINT
2000 PRINT
2010 PRINT
2020 PRINT
2030 CALL SOUND(120,392,0)
```

```
2040 CALL KEY(0,KY,ST)
2050 IF ST=0 THEN 2040
2060 CALL SOUND(350,131,0,392,0,1047,0)
2070 IF KY=57 THEN 2180
2080 IF KY=56 THEN 2200
2090 IF KY=55 THEN 2220
2100 IF KY=54 THEN 2240
2110 IF KY=53 THEN 2260
2120 IF KY=52 THEN 2280
2130 IF KY=51 THEN 2300
2140 IF KY=50 THEN 2320
2150 IF KY=49 THEN 2340
2160 IF KY=48 THEN 2360
2170 IF KY=83 THEN 2380 ELSE 2040
2180 Y=9
2190 GOTO 2400
2200 Y=8
2210 GOTO 2400
2220 Y=7
2230 GOTO 2400
2240 Y=6
2250 GOTO 2400
2260 Y=5
2270 GOTO 2400
2280 Y=4
2290 GOTO 2400
2300 Y=3
2310 GOTO 2400
2320 Y=2
2330 GOTO 2400
2340 Y=1
2350 GOTO 2400
2360 Y=0
2370 GOTO 2400
2380 Y=30
2390 GOTO 2400
2400 REM ...................
2410 REM
2420 REM     VOLUME DE LA
    BASSE
2430 REM
2440 REM ...................
2450 CALL CLEAR
2460 PRINT "  CHOISISSEZ LE  VOLUME DE
         LA BASSE"
2470 PRINT
2480 PRINT
2490 PRINT "           (0 A 9)"
2500 PRINT
2510 PRINT "   (S POUR LA SUPPRIMER)"
2520 PRINT
2530 PRINT
2540 PRINT
2550 CALL SOUND(120,131,0)
2560 CALL KEY(0,KY,ST)
2570 IF ST=0 THEN 2560
2580 CALL SOUND(350,131,0,1047,0,392,0)
2590 IF KY=57 THEN 2700
2600 IF KY=56 THEN 2720
2610 IF KY=55 THEN 2740
2620 IF KY=54 THEN 2760
2630 IF KY=53 THEN 2780
2640 IF KY=52 THEN 2800
2650 IF KY=51 THEN 2820
2660 IF KY=50 THEN 2840
2670 IF KY=49 THEN 2860
2680 IF KY=48 THEN 2900
2690 IF KY=83 THEN 2880 ELSE 2560
2700 W=9
2710 GOTO 2920
2720 W=8
2730 GOTO 2920
2740 W=7
2750 GOTO 2920
2760 W=6
2770 GOTO 2920
2780 W=5
2790 GOTO 2920
2800 W=4
2810 GOTO 2920
2820 W=3
2830 GOTO 2920
2840 W=2
2850 GOTO 2920
2860 W=1
2870 GOTO 2920
2880 W=30
2890 GOTO 2920
2900 W=0
2910 GOTO 2920
2920 CALL CLEAR
2930 PRINT "DES CORRECTIONS A APPORTER?
                                        (
O/N)"
2940 FOR F=1 TO 8
2950 PRINT
2960 NEXT F
2970 CALL KEY(0,KY,ST)
2980 IF ST=0 THEN 2970
2990 IF KY=79 THEN 730
3000 IF KY=78 THEN 3010 ELSE 2970
3010 REM ...................
3020 REM
3030 REM   MUSIC BOX DANCER
3040 REM
3050 REM ...................
3060 CALL CLEAR
3070 PRINT "    *********************"
3080 PRINT "    *  MUSIC BOX DANCER  *"
3090 PRINT "    *********************"
3100 FOR F=1 TO 5
3110 PRINT
3120 NEXT F
3130 PRINT " TEMPO:";Z;"MILLISECONDES"
3140 PRINT
3150 PRINT " VOLUME CHANT:";T
3160 PRINT
3170 PRINT " VOLUME ACCOMPAGNEMENT:";Y
3180 PRINT
3190 PRINT " VOLUME BASSE:";W
3200 PRINT
3210 FOR DEL=1 TO 800
3220 NEXT DEL
3230 REM
3240 FOR DEL=1 TO 3
3250 CALL SOUND(X,G,T)
3260 CALL SOUND(X,I,T)
3270 CALL SOUND(X,K,T)
3280 CALL SOUND(X,I,T)
3290 CALL SOUND(X,M,T)
3300 CALL SOUND(X,K,T)
3310 CALL SOUND(X,I,T)
3320 CALL SOUND(X,K,T)
3330 NEXT DEL
3340 CALL SOUND(X,G,T)
3350 CALL SOUND(X,I,T)
3360 CALL SOUND(X,K,T)
3370 CALL SOUND(X,I,T)
3380 CALL SOUND(X,M,T)
3390 CALL SOUND(X,K,T)
3400 CALL SOUND(X,I,T)
3410 FOR U=1 TO 2
3420 CALL SOUND(X,M,T,K,Y)
3430 CALL SOUND(X,M,T,G,Y)
```

```
3440 CALL SOUND(X,K,T,I,Y)
3450 CALL SOUND(X,M,T,K,Y)
3460 CALL SOUND(X,O,T,G,Y)
3470 CALL SOUND(X,M,T,I,Y)
3480 CALL SOUND(X,O,T,K,Y)
3490 CALL SOUND(X,Q,T,G,Y)
3500 CALL SOUND(X,M,T,K,Y)
3510 CALL SOUND(X,S,T,J,Y)
3520 CALL SOUND(X,988,T,L,Y)
3530 CALL SOUND(X,R,T,M,Y)
3540 CALL SOUND(X,Q,T,P,Y)
3550 CALL SOUND(X,Q,T,G,Y)
3560 CALL SOUND(X,Q,T,K,Y)
3570 CALL SOUND(X,K,T)
3580 CALL SOUND(X,M,T)
3590 CALL SOUND(X,Q,T,E,Y)
3600 CALL SOUND(X,P,T,247,Y)
3610 CALL SOUND(X,N,T,H,Y)
3620 CALL SOUND(X,494,T,K,Y)
3630 CALL SOUND(X,K,T,247,Y)
3640 CALL SOUND(X,494,T,H,Y)
3650 CALL SOUND(X,N,T,K,Y)
3660 CALL SOUND(X,P,T,H,Y)
3670 CALL SOUND(X,O,T,G,Y)
3680 CALL SOUND(X,M,T,I,Y)
3690 CALL SOUND(X,R,T,M,Y)
3700 CALL SOUND(X,Q,T,K,Y)
3710 CALL SOUND(X,Q,T,I,Y)
3720 CALL SOUND(X,Q,T,K,Y)
3730 CALL SOUND(X,M,T)
3740 CALL SOUND(X,M,T,K,Y)
3750 CALL SOUND(X,M,T,G,Y)
3760 CALL SOUND(X,K,T,I,Y)
3770 CALL SOUND(X,M,T,K,Y)
3780 CALL SOUND(X,O,T,G,Y)
3790 CALL SOUND(X,M,T,I,Y)
3800 CALL SOUND(X,O,T,K,Y)
3810 CALL SOUND(X,Q,T,G,Y)
3820 CALL SOUND(X,M,T,K,Y)
3830 CALL SOUND(X,S,T,J,Y)
3840 CALL SOUND(X,988,T,L,Y)
3850 CALL SOUND(X,R,T,M,Y)
3860 CALL SOUND(X,Q,T,P,Y)
3870 CALL SOUND(X,Q,T,G,Y)
3880 CALL SOUND(X,Q,T,K,Y)
3890 CALL SOUND(X,K,T)
3900 CALL SOUND(X,M,T)
3910 CALL SOUND(X,Q,T,E,Y)
3920 CALL SOUND(X,P,T,247,Y)
3930 CALL SOUND(X,N,T,H,Y)
3940 CALL SOUND(X,494,T,K,Y)
3950 CALL SOUND(X,K,T,247,Y)
3960 CALL SOUND(X,494,T,H,Y)
3970 CALL SOUND(X,N,T,K,Y)
3980 CALL SOUND(X,494,T,H,Y)
3990 CALL SOUND(X,M,T,G,Y)
4000 CALL SOUND(X,K,T,I,Y)
4010 CALL SOUND(X,O,T,K,Y)
4020 CALL SOUND(X,M,T,M,Y)
4030 CALL SOUND(X,M,T,I,Y)
4040 CALL SOUND(X,M,T,K,Y)
4050 CALL SOUND(X,M,T)
4060 CALL SOUND(X,M,T,E,Y)
4070 CALL SOUND(X,M,T,A,Y)
4080 CALL SOUND(X,K,T,E,Y,A,W)
4090 CALL SOUND(X,M,T,I,Y,A,W)
4100 CALL SOUND(X,O,T,I,Y,A,W)
4110 CALL SOUND(X,M,T,E,Y,A,W)
4120 CALL SOUND(X,O,T,E,Y,A,W)
4130 CALL SOUND(X,Q,T,I,Y,A,W)
4140 CALL SOUND(X,M,T,A,W)
4150 CALL SOUND(X,S,T,D,W)
4160 CALL SOUND(X,988,T,J,Y,D,W)
4170 CALL SOUND(X,R,T,G,Y,D,W)
4180 CALL SOUND(X,Q,T,E,Y,A,W)
4190 CALL SOUND(X,Q,T,A,Y,C,W)
4200 CALL SOUND(X,Q,T,E,Y,A,W)
4210 CALL SOUND(X,C,T,A,W)
4220 CALL SOUND(X,G,T,A,W)
4230 CALL SOUND(X,Q,T,E,W)
4240 CALL SOUND(X,P,T,K,Y,E,W)
4250 CALL SOUND(X,N,T,J,Y,E,W)
4260 CALL SOUND(X,494,T,J,Y,E,W)
4270 CALL SOUND(X,K,T,E,W)
4280 CALL SOUND(X,494,T,E,W)
4290 CALL SOUND(X,N,T,H,Y,E,W)
4300 CALL SOUND(X,P,T,K,Y,E,W)
4310 CALL SOUND(X,O,T,A,W)
4320 CALL SOUND(X,M,T,K,Y,A,W)
4330 CALL SOUND(X,R,T,I,Y,A,W)
4340 CALL SOUND(X,Q,T,I,Y,A,W)
4350 CALL SOUND(X,Q,T,E,Y,A,W)
4360 CALL SOUND(X,Q,T,E,Y,A,W)
4370 CALL SOUND(X,I,T,A,W)
4380 CALL SOUND(X,M,T,K,Y,A,W)
4390 CALL SOUND(X,M,T,A,W)
4400 CALL SOUND(X,K,T,E,Y,A,W)
4410 CALL SOUND(X,M,T,I,Y,A,W)
4420 CALL SOUND(X,O,T,I,Y,A,W)
4430 CALL SOUND(X,M,T,K,Y,A,W)
4440 CALL SOUND(X,O,T,K,Y,A,W)
4450 CALL SOUND(X,Q,T,I,Y,A,W)
4460 CALL SOUND(X,M,T,A,W)
4470 CALL SOUND(X,S,T,D,W)
4480 CALL SOUND(X,988,T,J,Y,D,W)
4490 CALL SOUND(X,R,T,L,Y,D,W)
4500 CALL SOUND(X,Q,T,K,Y,A,W)
4510 CALL SOUND(X,Q,T,K,Y,A,W)
4520 CALL SOUND(X,Q,T,G,Y,A,W)
4530 CALL SOUND(X,I,T,A,W)
4540 CALL SOUND(X,K,T,A,W)
4550 CALL SOUND(X,Q,T,E,W)
4560 CALL SOUND(X,P,T,K,Y,E,W)
4570 CALL SOUND(X,N,T,H,Y,E,W)
4580 CALL SOUND(X,494,T,H,Y,E,W)
4590 CALL SOUND(X,K,T,I,Y,E,W)
4600 CALL SOUND(X,494,T,K,Y,E,W)
4610 CALL SOUND(X,N,T,H,Y,E,W)
4620 CALL SOUND(X,494,T,K,Y,E,W)
4630 CALL SOUND(X,M,T,A,W)
4640 CALL SOUND(X,K,T,E,Y,A,W)
4650 CALL SOUND(X,O,T,G,Y,A,W)
4660 CALL SOUND(X,M,T,G,Y,A,W)
4670 CALL SOUND(X,M,T,K,Y,A,W)
4680 CALL SOUND(X,M,T,K,Y,A,W)
4690 CALL SOUND(X,M,T,I,Y,A,W)
4700 CALL SOUND(X,M,T,K,Y,A,W)
4710 V=0
4720 V=V+1
4730 CALL SOUND(X,S,T,P,T,M,T)
4740 CALL SOUND(X,R,T,J,Y,D,W)
4750 CALL SOUND(X,P,T,L,Y,D,W)
4760 CALL SOUND(X,M,T,L,Y,D,W)
4770 CALL SOUND(X,L,T,J,Y,D,W)
4780 CALL SOUND(X,M,T,J,Y,D,W)
4790 CALL SOUND(X,P,T,L,Y,D,W)
4800 CALL SOUND(X,R,T,K,Y,D,W)
4810 CALL SOUND(X,Q,T,A,W)
4820 CALL SOUND(X,M,T,K,Y,A,W)
4830 CALL SOUND(X,R,T,G,Y,A,W)
4840 CALL SOUND(X,Q,T,I,Y,A,W)
4850 CALL SOUND(X,Q,T,K,Y,A,W)
4860 IF V=2 THEN 5080
4870 CALL SOUND(X,Q,T,K,Y,A,W)
```

```
4880 CALL SOUND(X,I,T,A,W)
4890 CALL SOUND(X,K,T,A,W)
4900 CALL SOUND(X,Q,T,N,T,K,T)
4910 CALL SOUND(X,P,T,K,Y,E,W)
4920 CALL SOUND(X,N,T,247,Y,E,W)
4930 CALL SOUND(X,494,T,247,Y,E,W)
4940 CALL SOUND(X,K,T,I,Y,E,W)
4950 CALL SOUND(X,494,T,247,Y,E,W)
4960 CALL SOUND(X,N,T,H,Y,E,W)
4970 IF V=3 THEN 5130
4980 CALL SOUND(X,P,T,K,Y,E,W)
4990 CALL SOUND(X,Q,T,A,W)
5000 CALL SOUND(X,M,T,G,Y,A,W)
5010 CALL SOUND(X,R,T,O,Y,A,W)
5020 CALL SOUND(X,Q,T,O,Y,A,W)
5030 CALL SOUND(X,Q,T,M,Y,A,W)
5040 CALL SOUND(X,O,T,M,Y,A,W)
5050 CALL SOUND(X,O,T,S,Y,A,W)
5060 CALL SOUND(X,Q,T,J,Y,A,W)
5070 IF V=1 THEN 4720 ELSE 32767
5080 CALL SOUND(X,M,T,K,Y,A,W)
5090 CALL SOUND(X,M,T,I,Y,A,W)
5100 CALL SOUND(X,M,T,K,Y,A,W)
5110 V=V+1
5120 IF V=3 THEN 4900
5130 CALL SOUND(X,494,T,K,Y,D,W)
5140 CALL SOUND(X,M,T,G,Y,A,W)
5150 CALL SOUND(X,K,T,G,Y,A,W)
5160 CALL SOUND(X,O,T,I,Y,A,W)
5170 CALL SOUND(X,M,T,K,Y,A,W)
5180 CALL SOUND(Z,M,T,P,7,A,3)
5190 CALL SOUND(X,110,30)
5200 NEXT U
5210 CALL SOUND(Z,110,30)
5220 FOR F=1 TO 30
5230 PRINT
5240 NEXT F
5250 PRINT "  ENCORE UNE FOIS?  (O/N)"
5260 FOR F=1 TO 10
5270 PRINT
5280 NEXT F
5290 CALL KEY(0,KY,ST)
5300 IF ST=0 THEN 5290
5310 IF KY=78 THEN 5330
5320 IF KY=79 THEN 5380 ELSE 5290
5330 PRINT "          AU REVOIR"
5340 PRINT
5350 PRINT
5360 PRINT
5370 END
5380 CALL CLEAR
5390 PRINT "JE VOUS RAPPELLE LE CHOIX   Q
UE VOUS AVIEZ FAIT:"
5400 PRINT
5410 PRINT "TEMPO:";Z;"MILLISECONDES"
5420 PRINT
5430 PRINT "VOLUME CHANT:";T
5440 PRINT
5450 PRINT "VOLUME ACCOMPAGNEMENT:";Y
5460 PRINT
5470 PRINT "VOLUME BASSE:";W
5480 PRINT
5490 PRINT
5500 PRINT "SOUHAITERIEZ VOUS EFFECTUER D
ES MODIFICATIONS?  (O/N)"
5510 PRINT
5520 PRINT
5530 CALL SOUND(120,392,0)
5540 CALL KEY(0,KY,ST)
5550 IF ST=0 THEN 5540
5560 IF KY=78 THEN 3010
5570 IF KY=79 THEN 5580 ELSE 5540
5580 INPUT "TEMPO:                      (
1 A 4250 MILLISECONDES)?":Z
5590 PRINT
5600 PRINT "   ATTENTION LE VOLUME
   DECROIT DE 0 A 30!..        -------"
5610 PRINT
5620 INPUT "VOLUME CHANT                (
0 A 30)?":T
5630 INPUT "VOLUME ACCOMPAGNEMENT       (
0 A 30)?":Y
5640 INPUT "VOLUME BASSE                (
0 A 30)?":W
5650 X=Z/2
5660 GOTO 3010
```

Erratum

Vous avez certainement remarqué, dans le listing du programme "Edipage" publié dans le numéro 6, une ligne tout à fait étrange :

2000 *SGN CALL GOTO LOG & !

Cette superbe chose provient d'une fantaisie de l'interpréteur, certainement dûe à une micro-coupure de courant.

En fait, la ligne 2000 était :

2000 T$(L-2,C-2)=" "

Nous avons eu droit à une autre aberration un peu plus loin, mais cette fois moins embêtante, puisqu'il s'agissait uniquement de la juxtaposition des lignes 2130 et 2140. □

Des nouvelles du Club Ticazur

Le nouveau numéro de téléphone est le 32.63.16 (appels si possible entre 20 et 22H).

Le cours d'initiation au Basic de Jean Marin est constitué de dix programmes de six à sept Ko chacun, sur une seule cassette et non, comme je vous l'avais indiqué par erreur sur dix cassettes. Chaque programme représente bien deux heures de cours.

Chaque cours de Jean Marin sur l'utilisation de la Mini-mémoire fait l'objet d'un compte-rendu dactylographié et dont les photocopies sont mises à la disposition des membres du club.

Six comptes-rendu ont été établis à ce jour représentant une trentaine de pages.

Gérard Baroni
Président de Ticazur
Villa Héléna
Chemin des Bas-Campons
06480 La Colle-sur-Loup
Tél : 32.63.16

Routines Graphiques pour le Basic Etendu

Marianne Sutz

Pour faire suite à de nombreuses demandes de lecteurs ne disposant pas de l'éditeur/assembleur ou de la Mini-mémoire, mais désirant avoir accès aux routines graphiques publiées dans les précédents numéros de "99 Magazine", nous vous proposons une version modifiée de certaines d'entre elles.

Afin de ne pas perturber le fonctionnement du Basic Etendu, nous avons dû faire quelques sacrifices : le choix des couleurs est beaucoup plus réduit (une seule couleur pour tous les points "on" et "off") et le nombre de routines a été légèrement diminué.

EF

Cette routine remplace la routine EFFP des versions précédentes. Elle réinitialise le tampon TIMAGE (image virtuelle en mémoire vive CPU) et ne requiert pas d'argument. Pour l'appeler :

```
CALL LINK("EF")
```

PO

Remplace la routine POINT destinée à afficher un point aux coordonnées X-Y. Le passage des paramètres est réalisé au moyen de l'instruction *CALL LOAD*. Ainsi, la séquence :

```
CALL LOAD(9730,127,96)
CALL LINK("PO")
```

affiche un point au centre de l'écran.

LI

Trace une ligne entre deux points quelconques. Les coordonnées X1, Y1, X2 et Y2 sont passées à la routine par *CALL LOAD* :

```
CALL LOAD(9730,0,0,255,191)
CALL LINK("LI")
```

trace une diagonale allant de l'angle inférieur gauche à l'angle supérieur droit de l'écran.

LE

Affiche le caractère de code ASCII *N* à l'intersection de la ligne *L* et de la colonne *C*. *N* doit être compris entre 33 et 143 inclus et les caractères redéfinis par *CALL CHAR* sont autorisés.

```
CALL LOAD(9730,65,1,1)
CALL LINK("LE")
```

affiche un "A" dans le coin supérieur gauche de l'écran.

IM

Effectue le transfert de l'image virtuelle vers la mémoire vidéo. Le retour au Basic Etendu est assuré par la touche FCTN-3 (ERASE).

Quelques précisions

Couleur de l'écran

La couleur par défaut de l'écran est noire mais il est possible de la modifier en "pokant" une valeur à l'adresse 15895 (>3E17). cette valeur doit être comprise entre 0 (incolore) et 15 (blanc). Si elle est supérieure à 15, le code de couleur résultant sera égal à : **X AND 15** (voir "Les opérateurs logiques", 99 Magazine numéro 5).
Pour obtenir un écran bleu foncé :

```
CALL LOAD(15895,5).
```

Couleurs des points "on" et "off"

Les couleurs par défaut sont : jaune clair pour les points "on" et vert foncé pour les points "off". Pour modifier ces couleurs, il faut poker, à l'adresse 16206 (>3F4E), une valeur ainsi calculée :
code de la couleur pour les points "on" * 16 + code de la couleur des points "off".
Pour obtenir des points blancs sur fond noir :

```
CALL LOAD(16206,241) (15*16+1=241).
```

Mémoire disponible pour le Basic Etendu

Les routines en langage machine utilisent la section basse de l'extension 32Ko, et une partie de la section haute pour la routine *LE* et le tampon *TBASIC*, jusqu'à l'adresse >CD69. Il reste donc pour votre programme la mémoire vidéo et 12950 octets de mémoire vive CPU.
Le Basic Etendu utilisant la section haute de l'extension 32Ko en partant des adresses les plus fortes, un programme trop important peut être partiellement détruit par le tampon *TIMAGE*; il faudra donc prendre garde à ne pas dépasser (vers le bas) l'adresse >CD69. En revanche, le risque de destruction d'une routine est très faible puisque la routine *LE* (la seule qui soit vulnérable) est située en >A000.

Chargement du code objet

Avant tout, ceux d'entre-vous qui disposent de lecteurs de disquettes devront utiliser la commande *CALL FILES(1)* pour deux raisons :

- gain de place dans la mémoire vidéo pour le programme en Basic Etendu;
- moins de risque de perturbations des blocs d'accès aux périphériques, qui peuvent être endommagés lors de l'exécution de la routine *IM*.

Pour le chargement du code, il faut employer le programme joint à cet article, qui implante directement les routines en mémoire. Une fois cette opération exécutée, vous pouvez charger vos propres programmes.

Si vous disposez d'un lecteur de disquettes, vous pouvez dans ce cas bénéficier d'un chargement plus rapide puisque le code objet correspondant au fichier source listé pages suivantes se trouve désormais sur la disquette "Routines graphiques". Pour le charger :

```
CALL INIT
CALL LOAD("DSKx.OBJETXB")
```

□

```
        DEF   EF,PO,IM,LI,LE
        AORG  >2602
LIGNED  DATA  0      * 9730
LIGNEA  DATA  0      * 9732
CR7     DATA  0
CR8     DATA  0
TIMAGE  BSS   >1800
REGVDP  DATA  >0002,>0206,>03FF,>0403
        DATA  >0536,>0607,>0711,>FFFF
RVDPBA  DATA  >0000,>0200,>0320,>0400
        DATA  >0506,>0600,>0717,>FFFF
ETAT    EQU   >837C
RCLAV   EQU   >8375
VSBW    EQU   >2020
VMBW    EQU   >2024
VSBR    EQU   >2028
VMBR    EQU   >202C
KSCAN   EQU   >201C
VWTR    EQU   >2030

LI      MOVB  @LIGNED,R9
        SRL   R9,8
        MOVB  @LIGNEA,R7
        SRL   R7,8
        MOV   R7,@CR7
        MOVB  @LIGNED+1,R2
        SRL   R2,8
        CI    R2,191
        JGT   LIGNE8
        LI    R10,191
        S     R2,R10
        MOVB  @LIGNEA+1,R2
        SRL   R2,8
        CI    R2,191
        JGT   LIGNE8
        LI    R8,191
        S     R2,R8
        MOV   R8,@CR8
*
        CLR   R12
        LI    R5,1
        LI    R6,1
        S     R9,R7
        JLT   LIGNE1
        JMP   LIGNE2
LIGNE1  NEG   R5
        NEG   R7
LIGNE2  MOV   R7,R7
        JNE   LIGNE3
        SETO  R12
LIGNE3  S     R10,R8
        JLT   LIGNE4
        JMP   LIGNE5
LIGNE4  NEG   R6
        NEG   R8
LIGNE5  MOV   R9,R0
        MOV   R10,R1
        MOV   R1,R4
        SLA   R4,5
        SOC   R1,R4
        ANDI  R4,>FF07
        A     R0,R4
        ANDI  R0,7
        S     R0,R4
        LI    R3,>8000
        SRC   R3,0
        SOCB  R3,@TIMAGE(R4)
        C     R9,@CR7
        JNE   LIGNE6
        C     R10,@CR8
        JNE   LIGNE6
LIGNE8  JMP   FHPLOT
LIGNE6  MOV   R12,R12
        JLT   LIGNE7
        A     R5,R9
        S     R8,R12
        JMP   LIGNE5
LIGNE7  A     R6,R10
        A     R7,R12
        JMP   LIGNE5

EF      LI    R0,TIMAGE
EFFP1   CLR   *R0+
        CI    R0,TIMAGE+>1800
        JEQ   FHPLOT
        JMP   EFFP1
PO      MOVB  @LIGNED,R0
        SRL   R0,8
        MOVB  @LIGNED+1,R2
        SRL   R2,8
        CI    R2,191
        JGT   FHPLOT
        LI    R1,191
        S     R2,R1
        MOV   R1,R4
        SLA   R4,5
        SOC   R1,R4
        ANDI  R4,>FF07
        A     R0,R4
        ANDI  R0,7
        S     R0,R4
        LI    R3,>8000
        SRC   R3,0
        SOCB  R3,@TIMAGE(R4)
FHPLOT  B     @RETOUR

IM      LI    R1,TBASIC
        LI    R0,>300
        LI    R2,11520
        BLWP  @VMBR

        LI    R1,REGVDP
IRBM    MOV   *R1+,R0
        JLT   ITLUT
        BLWP  @VWTR
        JMP   IRBM

ITLUT   LI    R0,>1B00
        LI    R1,>D000
        BLWP  @VSBW

        LI    R0,>1800
ITE     MOV   R0,R1
        SWPB  R1
        BLWP  @VSBW
        INC   R0
        CI    R0,>1800+>300
        JNE   ITE

        LI    R0,>2000
        LI    R1,>BC00
TTC     BLWP  @VSBW
        INC   R0
        CI    R0,>3800
        JNE   TTC

        CLR   R0
        LI    R1,TIMAGE
        LI    R2,>1800
        BLWP  @VMBW

CLAV    BLWP  @KSCAN
        CB    @RCLAV,@REGVDP+12
        JNE   CLAV
```

```
RBIS    LI    R1,RVDPBA
IRB     MOV   *R1+,R0
        JLT   TTBAS
        BLWP  @VWTR
        JMP   IRB

TTBAS   LI    R0,>300
        LI    R1,TBASIC
        LI    R2,11520
        BLWP  @VMBW

        LI    R0,>2FF
        LI    R1,>8000
ESPACE  BLWP  @VSBW
        DEC   R0
        JOC   ESPACE

RETOUR  CLR   R0
        MOVB  R0,@ETAT
        RT

        AORG  >A000
LE      MOVB  @LIGNED,R4
        SRL   R4,8
        CI    R4,32
        JLE   L3
L1      LI    R5,1024
        CI    R4,143
        JH    L3
L2      AI    R4,-32
        SLA   R4,3
        A     R4,R5
        MOVB  @LIGNEA,R4
        SRL   R4,8
        CI    R4,32
        JH    L3
        DEC   R4
        JNC   L3
        SLA   R4,3
        MOVB  @LIGNED+1,R6
        SRL   R6,8
        CI    R6,24
        JH    L3
        DEC   R6
        JNC   L3
        MOV   R6,R6
        JEQ   L4
L5      AI    R4,256
        DEC   R6
        JNE   L5
L4      LI    R2,TIMAGE
        MOV   R5,R0
        A     R4,R2
        LI    R3,8
L7      BLWP  @VSBR
        MOVB  R1,*R2+
        INC   R0
        DEC   R3
        JNE   L7
L3      B     @RETOUR
TBASIC  DATA  0
        END
```

```
100 ! ***********************
110 ! *     Ce  programme   *
120 ! *  implante  le code  *
130 ! *  .   objet  dans    *
140 ! *  l'extension  32Ko  *
150 ! ***********************
160 ! *   Basic Etendu  +   *
170 ! *   extension  32Ko   *
180 ! ***********************
182 ! * CALL FILES(1) / NEW *
183 ! *   avant execution   *
184 ! ***********************
190 !
200 CALL INIT
210 !
220 ! Pointeur premiere
230 ! reference externe
240 !
250 CALL LOAD(8196,63,216)
255 !
260 ! References externes
270 !
280 DATA 76,69,32,32,32,32,160,0,76,73,32
,32,32,32,62,42,73,77,32,32,32,32,63,10
290 DATA 80,79,32,32,32,32,62,210,69,70,3
2,32,32,32,62,196
300 FOR I=16344 TO 16383 :: READ N :: CAL
L LOAD(I,N):: NEXT I
310 !
320 ! Code objet
330 !
340 DATA 0,2,2,6,3,255,4,3,5,54,6,7,7,17,
255,255,0,0,2,0,3,32,4,0,5,6,6,0,7,23,2
55,255
350 DATA 210,96,38,2,9,137,209,224,38,3,9
,135,200,7,38,6,208,160,38,3,9,130,2,13
0,0,191
360 DATA 21,54,2,10,0,191,98,130,208,160,
38,5,9,130,2,130,0,191,21,45,2,8,0,191,
98,2
370 DATA 200,8,38,8,4,204,2,5,0,1,2,6,0,1
,97,201,17,1,16,2,5,5,5,7,193,199,22,1,
7,12
380 DATA 98,10,17,1,16,2,5,6,5,8,192,9,19
2,74,193,1,10,84,225,1,2,68,255,7,161,0
,2,64
390 DATA 0,7,97,0,2,3,128,0,11,3,249,3,38
,10,136,9,38,6,22,4,136,10,38,8,22,1,16
,41
400 DATA 195,12,17,3,162,69,99,8,16,228,1
62,134,163,7,16,225,2,0,38,10,4,240,2,1
28,62,10
410 DATA 19,27,16,251,208,32,38,2,9,128,2
08,160,38,3,9,130,2,130,0,191,21,17,2,1
,0,191
420 DATA 96,66,193,1,10,84,225,1,2,68,255
,7,161,0,2,64,0,7,97,0,2,3,128,0,11,3,2
49,3
430 DATA 38,10,4,96,63,164,2,1,160,104,2,
0,3,0,2,2,45,0,4,32,32,44,2,1,62,10,192
,49
440 DATA 17,3,4,32,32,48,16,251,2,0,27,0,
2,1,208,0,4,32,32,32,2,0,24,0,192,64,6,
193
450 DATA 4,32,32,32,5,128,2,128,27,0,22,2
48,2,0,32,0,2,1,188,0,4,32,32,32,5,128,
2,128
460 DATA 56,0,22,250,4,192,2,1,38,10,2,2,
24,0,4,32,32,36,4,32,32,28,152,32,131,1
17
470 DATA 62,22,22,250,2,1,62,26,192,49,17
,3,4,32,32,48,16,251,2,0,3,0,2,1,160,10
4,2,2
480 DATA 45,0,4,32,32,36,2,0,2,255,2,1,12
8,0,4,32,32,32,6,0,24,252,4,192,216,0,1
31,124
```

```
490 DATA 4,91,209,32,38,2,9,132,2,132,0,3
2,18,44,2,5,4,0,2,132,0,143,27,39,2,36,
255,224
500 DATA 10,52,161,68,209,32,38,4,9,132,2
,132,0,32,27,29,6,4;23,27,10,52,209,160
,38,3
510 DATA 9,134,2,134,0,24,27,20,6,6,23,18
,193,134,19,4,2,36,1,0,6,6,22,252,2,2,3
8,10
520 DATA 192,5,160,132,2,3,0,8,4,32,32,40
,220,129,5,128,6,3,22,250,4,96,63,164
530 FOR I=15882 TO 16299 :: READ N :: CAL
L LOAD(I,N):: NEXT I
540 FOR I=-24576 TO -24473 :: READ N :: C
ALL LOAD(I,N):: NEXT I
```

Courrier des lecteurs

Alexandre Duback

Mon TI se bloque parfois spontanément, soit en cours de programmation, soit à l'exécution d'un programme, même avec un module comme celui des échecs. Ce blocage semble se produire lors de la frappe de <ENTER>. L'écran affiche alors des "dessins", barres ou autre, de différentes couleurs. Ou, tout simplement, le pointeur disparaît et il est impossible d'entrer quoi que ce soit. Cet incident se manifeste même parfois à l'allumage de la console. Existe-t-il un remède ou une explication ?*

Lucien Gruny - 8, rue Vitalis - 13005 Marseille

Votre problème provient certainement d'une mauvaise liaison entre les modules et la console. Vous pouvez essayer de nettoyer les contacts placés sur le connecteur des modules avec un coton-tige imbibé d'alcool. Si les pannes persistent, il faudra faire changer le connecteur par le service après-vente.

Dans le catalogue d'EASY BUG (Mini-mémoire), il existe deux options pour enregistrer ou charger des programmes en Assembleur sur cassette. C'est bien joli mais, lorsque l'on possède un lecteur de disquettes, comment fait-on pour enregistrer des programmes ?

L'option "LOAD and RUN" permet de charger des programmes, mais auparavant il a bien fallu les enregistrer !

Avec la Mini-mémoire, il n'est pas possible de transférer simplement des codes objet vers une disquette. Si vous disposez de l'éditeur/assembleur et du désassembleur publié dans le numéro 5 de "99 Magazine", vous pouvez employer la méthode suivante :

- charger le programme depuis la cassette avec *EASY BUG*, option *L*;
- charger le désassembleur;
- désassembler le programme à transférer en indiquant un fichier sur disquette comme destination;
- charger l'assembleur et réassembler le programme, qui se trouvera dorénavant sur disquette.

Si vous ne possédez pas l'éditeur/assembleur, il faudra écrire une routine de transfert ou attendre celle que nous publierons prochainement.

Le désassembleur en assembleur que vous avez publié dans le numéro 5 de "99 Magazine" comporte une erreur (pas bien grave !) très facile à corriger. A la troisième ligne du sous-programme appelé FVIII1 (page 28, en haut de la colonne de gauche), il faut remplacer l'instruction MOV @REGPP+4,R4 par MOV @REGPP+8,R4. Ainsi, les instructions STWP et STST sont désassemblées correctement, alors que l'opérande trouvé était toujours R12, quelle que soit la valeur correspondante dans le code à désassembler.

On peut aussi en profiter pour intervertir les commentaires précédant les sous-programmes FVIII2 et FVIII1, qui ne sont pas à leur place.

C'est en utilisant le programme pour désassembler les 8Ko de la ROM moniteur (ce qui ne prend guère plus de trois minutes...) que j'ai remarqué cette erreur. J'espère que ce rectificatif profitera aux "bricoleurs de système", pour lesquels ce désassembleur est un outil essentiel.

Jean Penné - 26, rue de la Chasse - 93130 Noisy le Sec.

Nous vous remercions et vous félicitons pour la découverte de ce "bug" qui nous avait échappé.

J'ai une petite remarque à vous faire et quelques questions à vous poser au sujet de votre programme de désassemblage.

Quand on désassemble un programme ou une routine sur fichier disquette, les mnémoniques apparaissent en sixième position sur la ligne, les cinq premières étant réservées à l'étiquette plus un espace. Si l'on désire à ce moment là incorporer une étiquette de six caractères, il faut décaler la ligne de 2 espaces, le reste du source se trouvant alors décalé d'autant. Je sais que c'est sans importance pour le réassemblage, mais ce n'est pas très beau ni bon pour la lisibilité.

Je pense que vous recherchez la perfection dans vos programmes et cela méritait d'être signalé. J'ai effectué les modifications nécessaires au niveau du tampon TLIGNE; il y a largement la place

pour pouvoir décaler les mnémoniques de deux espaces vers la droite, sans toucher à la position des autres données.

Je travaille absolument seul et mes seules références sont : la documentation Texas (en anglais !), l'initiation à l'assembleur de Denise Amrouche et vos programmes qui sont pour moi une véritable mine d'or.

Je "décortique" en ce moment vos programmes "MASTERMIND" et "DESASSEMBLEUR". C'est absolument fantastique les astuces que je découvre; tous les jours j'en trouve de nouvelles ! Si la totalité du Mastermind me paraît claire, il y a deux choses qui me chiffonnent dans le désassembleur. En effet, pourquoi réserver 54 octets pour le bloc d'accès aux périphériques alors que le plus grand mot à écrire ne peut dépasser 20 caractères pour une imprimante série avec quelques paramètres ? Ce qui fait, avec les dix octets réservés aux arguments du fichier, un maximum de 30 octets. Y aurait-il possibilité de rajouter d'autres paramètres ?

Je cherche en ce moment à écrire sur imprimante une ligne tirée d'un fichier disquette, comment passer les paramètres pour PIO (caractères compressés, dilatés, etc...).

Serait-ce dans cette partie libre du PAB où à partir d'un tampon quelconque ?

Dans la routine d'ouverture du fichier, vous allez chercher une adresse dans la GROM/GRAM en >9802 immédiatement après avoir chargé la longueur du nom du périphérique utilisé, adresse qui réapparaît dans la routine RETOUR. Je n'arrive pas à déterminer la nature de cette adresse dans les explications anglaises; qu'en est-il exactement ?

Je constate dans vos articles que vous vous centrez beaucoup sur l'utilisation de routines en assembleur à partir du Basic, et rarement à partir du Basic Etendu. Ce module poserait-il certains problèmes.

Pour conclure, je ne manquerais pas de vous dire que votre revue est pour moi quelque chose d'indispensable, n'ayant aucune autre source d'information sérieuse à me mettre sous la dent.

Michel Teyssou - Arbus - 64230 Lescar.

Au sujet de la position des codes mnémoniques, il y a plusieurs explications au choix que nous avons fait :

- Si l'on décale les codes de deux positions, chaque enregistrement sur la disquette est augmenté d'autant. Cela n'est pas grave pour un petit fichier mais, pour 500 enregistrements, cela fait 1000 octets de plus...
- Le tampon utilisé pour l'affichage sur l'écran étant le même que celui utilisé pour les périphériques, il n'était pas possible de décaler les mnémoniques sans perdre d'autres informations à l'écran, à moins d'écrire une routine de modification du tampon selon l'organe de sortie, ce qui était peut être une complication inutile.
- L'occupation du tampon étant calculée au plus juste, il est fort probable que, si vous décalez les codes mnémoniques sans décaler d'autant les adresses, la fin d'une instruction utilisant le mode d'adressage symbolique indexé par un registre supérieur à 9 pour les deux opérandes (MOVB @>A000 (R10) ,@>A002 (R11), par exemple) sera écrasée par l'adresse de l'instruction.

Le bloc d'accès aux périphériques n'est pas vraiment surdimensionné. Par exemple, si nous prenons :

```
RS232/2.BA=9600.DA=8.PA=E.TW.NU.CH
.EC.CR.LF
```

cela nous fait 43 octets plus les 10 octets pour les arguments du fichier, soit 53 caractères. Le 54ème octet est seulement là pour obtenir un nombre pair (pas indispensable en mémoire vidéo mais c'est une bonne habitude avec un processeur 16 bits).

IL nous est impossible d'analyser la gestion des périphériques en assembleur dans le courrier des lecteurs. Par contre, vous trouverez dans le prochain numéro un article détaillé sur ces techniques.

Lorsque l'on vient du Basic (ou du Basic Etendu), il est bon de revenir à la bonne adresse. L'interface RS232 modifie cette adresse; il faut donc la sauvegarder avant d'adresser le périphérique en question. L'adresse >9802 (GROM Read Address Register) pointe sur notre adresse de retour et est auto-incrémentée, tout comme les adresses de lecture et écriture en mémoire vidéo.

```
MOVB @>9802,@RET    * sauvegarde le
                      poids fort de
                      l'adresse
MOVB @>9802,@RET+1  * sauvegarde le
                      poids faible de
                      l'adresse
DEC @RET  * puisque >9802 est auto-
            incrémentée, on décrémente
            @RET de 1 pour obtenir
            l'adresse réelle.
```

Le Basic Etendu ne fait pas toujours bon ménage avec l'assembleur : certaines routines n'existe pas (DSRLNK par exemple) et le Basic Etendu utilise la partie haute de l'extension 32Ko. Ceci explique notre préférence pour le Basic TI. Malgré tout, vous trouverez dans ce numéro une version "Basic Etendu" des routines graphiques.

Petites annonces (gratuites)

Carte P-Code et compilateur Pascal (02/84) vendu 2600 F ou échange contre imprimante GP100A avec câble TI-99/4A. M. Allot - 169, cours Emile Zola -69100 Villeurbanne - 16 (7) 885.00.77 - heures de bureau.

Vends TI-99/4A, Basic Etendu, Mini-mémoire, Parsec, Chisholm Trail, Carwars, Tombstone City, Return to pirate isle, câble cassettes. 3650 F. (1) 655.10.12 après 18H.

Cherche personne qui pourrait cédé synthétiseur de paroles (PHP 1500) avec, si possible, le module "Speech editor, ou à bas prix. Thierry Hocreitere - 23, rue Paul Schleiss - 51100 Reims.

Recherche plan de mémoire vive pour TI-99/4A. Philippe Escaich - Lescure - 09420 Rimont.

Recherche module "Adventure" et "Echecs". Frédéric Sabatier - Durandeau - 33440 Ambares - (56) 38.94.21.

Vends TI-99/4A (12/83), manettes de jeux, câble magnéto, "Munch Man", "Echecs", 12 livres, 10 cassettes programmes. 2000 F. Christophe Rappeneau - 107, Avenue du 8 mai 1945 - Coulanges les Nevers - 58640 Varennes-Vauzelles.

Vends TI-99/4A, modulateur SECAM/UHF, magnéto compatible TI (thomson avec compteur), câble magnéto, manettes de jeux, sept livres pour TI, six modules dont Parsec, Tombstone City, etc..., cassettes jeux rétro I et II, synthétiseur de paroles, module Basic Etendu avec cassette "Basic Etendu par soi-même". Prix à débattre. (94) 94.32.77 H.R.

Vends TI-99/4A, boîtier d'extension, carte 32 Ko, carte RS232, Basic Etendu, synthétiseur de paroles, manettes, câble magnéto, quatre modules de jeux. L'ensemble : 8000 F. Jean-Louis Borel - le Val Romain A - Impasse Romain - 83000 Toulon - (94) 31.01.07 après 20h.

Vends TI-99/4A, cassette Basic, modules Foot, Invaders, Parsec, cordon magnéto. 1800 F. René Wiard - 13, Quartier Pichard - Appartement 5011 - 93700 Drancy. (1) 831.30.70.

Recherche carte contrôleur, lecteur de disquettes intégré, synthétiseur et éditeur de paroles. Alain Couegnat - 66, rue du Repos - 69007 Lyon. 16 (7) 858.21.16.

Vends TI-99/4A, module "Basic Etendu", Mini-mémoire, Echecs, manettes de jeux, livres sur TI-99, programmes sur cassettes. 4000 F. Georges Mouchot - 159 La Chaud - 43120 Monistrol sur Loire.

Urgent : vends Mini-mémoire + Manuel 200 pages en français (875 F), Burger Time, Speech Editor (215 F). TI Inveders, Carwars, Zerozap (160 F), Music Maker (275 F), 2 cassettes TI (90 F), livres et revues (5*17 F). Calogero - 28, lot Le Gat - 31490 Pibrac. (61) 86 02 75.

Echange carte 32 Ko neuve avec documentation contre TI Writer ou autre logiciels ou accessoires. René Giustini - 58, rue des Varennes - 45650 St Jean le Blanc. (38) 66.21.39.

Cherche module "Aventure". François Bronnec - la Croix des Essarts - Saint-Roch - 44160 Pont Chateau.

Achète Editeur/Assembleur, Pascal, Speech Editor, TI Logo... Etudie toutes propositions. Gérard Diebold - 27, Avenue Lacassagne - 69003 Lyon - (7) 233.61.06.

Vends TI-99/4A (2/83), câble cassettes, nombreux livres et revues, lecteur de cassettes, manettes, très bon état. 1100 F. Nombreux jeux, prix à la carte. J. Planté. (1) 916.09.60.

Cherche Editeur/Assembleur avec manuel pour TI-99/4A. Docteur R. Savigny - 26, rue du Colonel de Marmare - 31240 L'Union. (61) 74.32.02.

Vends module de jeu "Parsec", 200 F. Pierre Druet - 58, rue Chateau-Jeannot - 57100 Thionville.

Cherche modules "Basic Etendu", "Parsec". Vends ou échange module "Tombstone City" et cassette "Basic Etendu par soi-même". Philippe Menevis - 6, rue des Muriers - 31500 Toulouse. (61) 20.57.51.

Vends TI-99/4A, manettes, cordon K7, carte 32Ko, Basic Etendu + manuel français, Logo, Munch Man, Echecs, cassettes (Basic par soi-même, aide à la programmation, conseil financier, jeux rétros I et II, jeux d'entreprise, 99 Magazine N° 2 et 3), livres (Initiation à l'assembleur, pratique de l'ordinateur familial N° 2, jeux et programmes N° 3, exercice TI, 5 numéros de 99 Magazine, schéma technique). Le tout 4800 F (valeur 6400 F).
Achète contrôleur et lecteur de disquettes et carte RS232. (89) 55.09.49 après 18h.

Recherche contrôleur et lecteur de disquettes ainsi que le manuel d'accompagnement. Christian Faye - 6, Avenue du Maréchal Lyauthey - 94130 Nogent sur Marne. 873.67.29 (domicile) 346.30.11 (bureau).

Achète, bon prix, carte contrôleur de disquettes, ou cherche possesseur pour renseignements et duplication. Possède solides connaissances en électronique. Cherche aussi photocopies du manuel du "Terminal emulator II" (1 Franc/page). Docteur M. Taoufiq - 112, résidence Guy Mollet - 59221 Bauvin. (20) 86.42.61.

Achète TI Logo I ou II et Early Logo learning fun. Faire offre à Béguin - route nationale 96 - Napollon - 13400 Aubagne. (42) 03.73.60.

Vends Mini-mémoire avec manuel français, opération fromage (115 F), Burger Time (200 F), Moon Sweper (200 F), Speech editor (200 F), TI Invaders, Car wars, Zerozap (170 F), 2 K7 (80 F), Musik maker (299 F), livres et revues (5/18 F). Monsieur Calogero - 28, lot le Gat - 31490 Pibrac. (61) 86.02.75. Urgent.

Vends extension mémoire Socrelec 16Ko (adresses >B800 à >F7FF) utilisable sans restriction avec l'assembleur de la Mini-mémoire (et aussi avec le Basic Etendu, mais sans cassettes) 3000 FB ou 450 FF. Georges

Dejolier - 20, rue Mercator - B-1070 Bruxelles. (32.2) 522.58.18.

Vends boîtier périphérique, carte RS232, carte 32Ko. Le tout état neuf, jamais servi. 3800 F port compris. (26) 82.69.48 ou (26) 02.69.48 après 18 h.

Vends Mini-mémoire (750 F), TI Logo II (750 F), carte RS232 + Terminal Emulator II (2600 F), Synthétiseur vocal + Speech Editor (1800 F), Parsec (200 F), Munch Man (200 F), Cars Wars (200 F), Blasto (200 F), Tombstone City (150 F). Bruno. (1) 534.85.13.

Vends TI-99 Juin 83, manettes de jeu, cordon magnéto, Basic par soi-même, Indoor Soccer, Alligator Mix. 2000 F. Bernard Cardon - 9, rue Félix Cadras - 62640 - Montigny-en-Gohelle. (21) 75.23.51.

Recherche plans pour RS232, pour liaison entre TI-99/4A et MODEM. Didier Guillon - 2, rue du 8 mai 1945 - 38190 Crolles.

Vends TI-99/4A, lecteur et contrôleur de disquettes, synthétiseur de parole, carte 32Ko, 10 logiciels (dont assembleur et Basic Etendu). 7000 F. Alexandre Goldszal - 9, rue de Douai - 75009 Paris. 280.33.43.

Recherche boîtier d'extension, carte mémoire 32Ko, module Editeur / Assembleur, synthétiseur de paroles, module Speech Editor. A. Winners - Fawetay 27 - 4804 Jalhay - Belgique.

Cherche module "Echecs" en français, module "Music Maker", synthétiseur de parole, cassette "Le Basic Etendu par soi-même". Henri Espi - Résidence "le Poséïdon" - 66750 Saint Cyprien Plage. (68) 21.22.12 le soir.

Vends TI-99/4A, boîtier d'extension, carte mémoire 32Ko, interface RS232, contrôleur de disquettes, lecteur de disquettes, programmes divers. Matériel état neuf. Prix à débattre. (1) 751.42.66 après 20H30.

Vends TI-99/4A avec magnéto Texas, manettes de jeux, modules "Munch Man", "Hustle", "Alpiner", "Blasto" et "Cars Wars" ainsi qu'un livre de programmes et une cassette "99 Magazine". Acheté 12/83. 2000 F. 353.10.15 après 17H30.

Recherche module Basic Etendu, Extension 32Ko, interface et programme pour faire du TELETYPE et de la TELEGRAPHIE avec TI-99. Monsieur Mismetti - La Tour - 73140 St Martin la Porte. (79) 56-65-64.

Achète RS232, modules "TI-Writer" et "Terminal Emulateur II", synthétiseur de paroles. Patrice Jean - 8, place du Chemin de Ronde - Appartement 200 - 78340 Les Clayes sous Bois. 055.28.65 après 19H00.

Vends 3900 F : TI-99/4A avec 10 modules (dont Parsec et Gestion de fichiers), module Basic Etendu, cordon magnéto et manettes de jeux. Avec 14 livres (dont assembleur) et 2 cassettes. (1) 656.22.54. Vends imprimante Mannesman Tally (liaison série). 3000 F. S'adresser au journal, qui transmettra.

Vends TI-99/4A, boîtier d'extension, carte 32Ko, contrôleur et lecteur de disquettes, Assembleur, Basic Etendu, manettes de jeux, cordon magnéto, Echecs, Parsec, livres (initiation à l'assembleur, jeux pour le TI), disquettes (Routines graphiques, Désassembleur, programmes divers), cassettes (rubis sacré, introduction aux jeux). 9500 F à débattre. 457.71.91 après 18h.

Cherche Editeur/Assembleur, Mini-mémoire, Terminal Emulator II et, éventuellement, d'autres modules. Alain Crozet - 15, rue B. Palissy - 16340 l'Isle d'Espagnac. (45) 69.02.96.

Cherche module TI Writer et interface série Texas Instrument. Faire offre au journal, qui transmettra.

99 MAGAZINE

BULLETIN D'ABONNEMENT

Je désire recevoir pour le N° ☐ de 99 magazine

☐ le numéro avec cassette 95 F TTC

☐ le numéro sans cassette 40 F TTC

☐ la cassette seule 55 F TTC

Je désire m'abonner pour 4 numéros

à partir du N° ☐

☐ avec cassette 325 F TTC

☐ sans cassette 135 F TTC

Nom ..

Adresse ..

..

Ces tarifs comprennent l'envoi postal en France Métropolitaine et CEE (voie aérienne exceptée)
Envoi par avion : + 40 F pour un abonnement - + 10 F pour un numéro seul

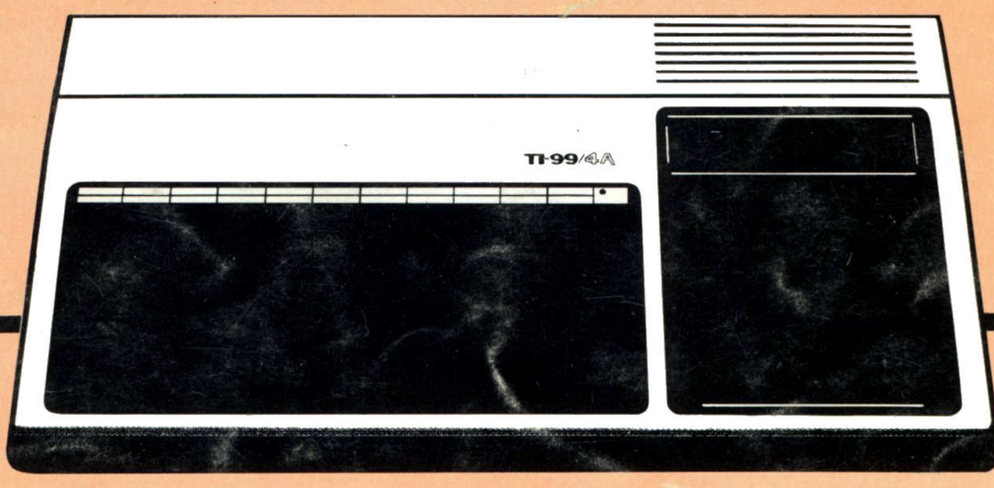
Envoyez ce bon de commande et votre règlement à :

Editions MEV - 64-70, rue des Chantiers - 78000 Versailles

99 MAGAZINE

CASSETTE D'ACCOMPAGNEMENT DISPONIBLE

Crayon optique
Mission Spéciale
Comptes familiaux
Bit-Map et Mini-mémoire



Numéro 8 - Trimestriel - Mars 1985 - 40 F

ISSN 0759-9005

Sommaires des numéros 1 à 7 de "99 Magazine"

Numéro 1

Musik Maker à l'essai	
Pourquoi acheter un TI-99/4A	
A vous de programmer	
Déballage du TI-99	
Les avantages du Basic Etendu	BE
Initiation au Basic	BTI
Jeu du Pendu	BTI
Fabriquez votre joystick	
Appelez moi KEY	BTI/BE
Programmes d'aide à la décision	BE
Jeu de l'Alphabet	BTI
Le club d'utilisateur TI-Education	
Introduction au LOGO	L
Les tours de Hanoï	BTI
Récupérez les programmes Apple	BTI/BE
Le troisième menuet de Back	BTI
Le dernier robot	BTI
Cherchez l'ancêtre	BTI
Gestion Personnelle de Fichiers	
Quand le TI devient un harmonium	BTI
Fichiers en Basic	BE

Numéro 2

Micro-monde LOGO : les lutins	L
La bataille navale	BTI
Le crocodile savant	
Jeu du Simon	BTI/BE
Jeu des Carrés	BTI
L'ordinateur de l'avenir	
TI-Writer à l'essai	
Canon de Purcell	BE
Chicken Helper	BE
Sauvegarde des fichiers et programmes	BTI/BE
Le Mastermind	BTI
Othello	BTI
Astéroïdes	BE
Kermit la grenouille	BE
Le Biorythme	BTI

Numéro 3

Un programme de calcul	BE
Annonce de Texas Instruments	
Othello à l'essai	
Apprendre le Basic en Programmant (1)	BTI
Introduction au Pascal	P
Puissance 4	BE
Les modes d'adressage en assembleur	A
Géographie : la carte de France	BTI
Créez votre touche RESET	
Puzzle	BTI
Awari	BTI
Le sapin de Noël	BE
PARSEC à l'essai	
Logo, seulement un langage pour enfants ?	L
Le Mastermind en assembleur	A
Du nouveau sur les cassettes	
Le Magnétophone TI	

Numéro 4

SOS Hélicoptère	BE
Utilisation de la Mini-mémoire	BTI
Communiqué de Texas Instruments France	
Le serveur de restaurant	BTI
"Echecs" à l'essai	
Combat de chars	BTI
Réalisez votre cordon de magnétophone	
Apprendre le Basic en programmant (2)	BTI
Jeu de mémorisation	BTI
Jack-pot	BE
Les nombres en Logo	L
Graphisme en haute résolution	BTI + A
Sous-programmes en assembleurs	A
Horloge	BE
Un détecteur de lumière pour le TI-99	
Le club M. T. I.	
Le système P-UCSD	P
Les lutins du Basic Etendu	BE
Le solitaire	BTI
Claustrophobia	BTI
"DISPLAY" & "ACCEPT" avec le module "PRK"	BTI
Une erreur dans le manuel de la Mini-mémoire	
Réparez votre assembleur ligne par ligne	A

Numéro 5

Conversions	BTI
Systèmes linéaires	BTI
Structure des disques du P-système	P
Poèmes et graphismes	BTI
Un désassembleur en assembleur	A
Les tours de Hanoï	BE
Trouvez le bon mot	BTI
Le jeu de la vie	BTI ou A
Programmathèque	
Les opérateurs logiques	BE
Apprendre le Basic en programmant (3)	BTI
Bibliographie	
Warangal	BTI
Graphismes en haute résolution	BTI + A
Le trésor de l'Armada	BE
Ecriture littérale d'un nombre	P
Copie d'écran sur imprimante Epson	BTI
Description du port d'entrées/sorties	
Translet	BE
Le club Ordinabis	
Le club Ticazur	

Numéro 6

Des chiffres et des lettres	BTI
A l'heure du Pascal	P
Kong	BE
Mini-mémoire moins "mini"	
Edipage	BTI
Caractères ASCII dans la page graphique	BTI + A
Musik	BTI
Copie d'écran sur Seïkosha GP-100	BE
Nouvelles fonctions avec la Mini-mémoire	BTI + A
Le Basic à la loupe	BTI & BE
Star treck	BE
Ecriture directe dans les registres VDP	BTI
My tailor was rich	BTI
Le code de la route	BE
Un double face sur votre TI	

Numéro 7

La défausse	BTI
Le tableur TI-CALC	
Sauvegarde d'images graphiques et copie d'écran	A
Copie d'écran sur Seïkosha GP-250	BE
La compilation séparée et les bibliothèques	P
Trouvez votre chemin dans les jeux d'aventures TI	A
Srand	BTI
Galaxia	A
LOGO et la récursivité	L
Dessinez avec le Basic étendu	BE
Accès au mode BIT-MAP en Basic TI	BTI
Music Box Dancer	BTI
Des nouvelles du club Ticazur	
Routines graphiques pour le Basic étendu	BE + A

* **BTI** : Basic TI. **BE** : Basic Etendu. **A** : Assembleur. **P** : Pascal. **L** : Logo.

Tous les numéros sont disponibles.

99 MAGAZINE

SOMMAIRE

5 — Editorial — Hervé Thiriez

6 — Bit-Map et Mini-mémoire — Gérard Baroni, Jean Marin

10 — Le train sifflera trois fois... — Denise Amrouche, Gilbert Arribet, Jean Sahal

15 — Comptes familiaux — Jean-philippe Guillemant

23 — Mission spéciale — Edouard Krok

29 — Crayon optique — Bernard Bezuel

54 — Le traitement des fichiers en Pascal — Gérard Santraille

60 — Editeur de caractères — Julien Thomas

61 — Kaleïdoscope — Marianne Sutz

62 — Opérations — Albert Rigal

65 — Petites annonces... gratuites

67 — Courrier des lecteurs — Alexandre Duback

Editions MEV - 64/70, rue des Chantiers - 78000 Versailles

Editorial

99 Magazine est-il toujours publié ?... c'est une question que l'on nous pose très souvent au téléphone, ou lors de salons tel le récent Micro-Expo. Ceci est certainement dû au fait que de nombreuses personnes pensent que nous avons un lien de parenté avec Texas Instruments, ce qui les laisse croire que "99 Magazine" a cessé d'exister au moment de l'arrêt de production du TI-99/4A. Quoi qu'il en soit, notre revue est bien présente et vous êtes toujours aussi nombreux à nous être fidèles, ce qui prouve qu'il est possible de publier une revue spécialisée bien après la fin de commercialisation d'un matériel. Passons maintenant au contenu de ce numéro.

***Gérard Baroni** et **Jean Marin**, nos deux amis du Club Ticazur, travaillent toujours avec acharnement sur la Mini-mémoire et nous confient le résultat de leurs investigations. **Bernard Bezuel**, quant a lui, nous propose un programme exceptionnel puisqu'il s'agit, à notre connaissance, du premier programme graphique pour TI-99 fonctionnant avec un crayon optique. Vous trouverez dans cet article les instructions et schémas permettant la réalisation du crayon. Les sources comportent de très nombreuses remarques fort intéressantes et nous vous conseillons vivement d'étudier la routine "REMPLISSAGE", remarquable...*

***Denise Amrouche**, **Gibert Arribet** et **Jean Sahal** ont regroupé leurs efforts afin de vous communiquer un programme où les possibilités sonores et graphiques du Basic Etendu sont pleinement exploitées. Vous connaissez tous la qualité des programmes écrits par **Edouard Krok**; il récidive avec le programme "Mission spéciale", dans lequel il vous faudra affronter l'abominable Krapul et ses acolytes.*

*Vous étiez nombreux à nous demander un programme de gestion de comptes familiaux; c'est désormais chose faite. Celui-ci a été écrit par **Jean-Philippe Guillemant** et fonctionne avec un lecteur de disquette et une extension 32Ko. Encore une fois, nous vous conseillons l'analyse du programme, écrit de façon remarquable. Il faut signaler que, sur la cassette d'accompagnement, le programme est scindé en deux parties qu'il vous faudra relier avec la classique instruction MERGE.*

***Gérard Santraille** poursuit sa série d'initiations au Pascal (on ne s'en lasse pas...) fort appréciée pour sa clarté et sa justesse par les amateurs de ce langage de haut niveau. "Opérations" est un programme a caractère éducatif, écrit en Basic TI par **Albert Rigal**.*

***Julien Thomas** nous propose un programme permettant la définition de caractères ou lutins et, pour conclure, **Marianne Sutz** vous communique un très court programme pour démontrer que le TI autorise des effets graphiques intéressants avec seulement quelques lignes de Basic.*

Hervé Thiriez

Rédacteur en chef - Directeur de la publication : Hervé Thiriez. **Comité de rédaction** : Jean-Luc Bazanegue, Gérard Santraille. **Dessins** : Laurent Bidot. **Ont collaboré à ce numéro** : Denise Amrouche, Gilbert Arribet, Gérard Baroni, Jean-Luc Bazanegue, Bernard Bezuel, Alexandre Duback, Jean-Philippe Guillemant, Edouard Krok, Jean Marin, Albert Rigal, Jean Sahal, Gérard Santraille, Marianne Sutz, Julien Thomas.

Editions MEV - 64/70, rue des Chantiers - 78000 Versailles.

Publicité : consultez les Editions MEV.

Impression : S.N.E. Repro-Versailles - 21, rue Danièle Casanova - 78210 St Cyr l'école - Tél. : 460.38.80.

Bit-Map et Mini-mémoire

Gérard Baroni et Jean Marin

N.D.L.R. : les programmes proposés avec cet article ne fonctionnent pas si un contrôleur de disquettes est raccordé au TI.

Dans ce troisième article, nous rappelons que notre but est de rendre compatible le mode Bit-Map - dont on connaît la puissance graphique - et le Basic dont on apprécie la souplesse d'édition.

La configuration minimale exigée pour mettre en œuvre les programme décrits dans cet article est : console, mini-mémoire et lecteur de K7.

Après l'écriture directe dans les registres VDP et l'accès au mode Bit-Map en Basic (14 Ko pour l'image et 2 Ko pour le Basic) et comme nous vous l'annoncions dans le dernier article, nous mettons à votre disposition les listings de routines en assembleur développées dans le cadre des activités du **Club TICAZUR.**

Ces routines, à implanter dans votre mini-mémoire, seront appelées du Basic par des CALL LINK ("BITMAP"), CALL LINK ("COLOR"), CALL LINK ("POINT",Y,X), CALL LINK ("DROITE", YD,XD,YA,XA).

Vous avez déjà compris, avec ces noms évocateurs, que vous allez pouvoir travailler sur des graphismes haute résolution avec toute la puissance et la vitesse d'exécution du langage machine et, toujours selon notre principe, avec la facilité d'emploi et d'édition du Basic. Désormais, les 2 Ko disponibles en Basic sont totalement libérés et utilisables pour des calculs, saisies, etc...

Les programmes dont les listings sont joints démontrent qu'en quelques lignes Basic vous pouvez arriver, rapidement, à des applications graphiques élaborées.

Quelques remarques sur ces programmes

LISSAJOUS

Comparez sa vitesse d'exécution et sa place en mémoire avec le programme Basic que nous vous avions proposé dans le numéro 7 de 99 Magazine.

KALEIDOSCOPE

Il vous est parfaitement possible de modifier ce programme et d'ajouter ou retrancher des miroirs. Vous ralentirez ou accélèrerez l'exécution selon votre option.

POLYGONE

Exemples :

- un angle de 10° donne un cercle;
- un angle de 150° donne une étoile à 12 pointes;
- un angle de 85° donne une couronne dont vont trouverez, joint à cet article, une copie d'écran sortie sur imprimante Seikosha GP100A.

LINES

Voici une version Basic du programme de démonstration "LINES" fourni par Texas Instruments avec la Mini-mémoire.

Nous ne pouvons pas, dans un article tel que celui-ci, commenter les routines et les programmes fournis comme nous le faisons à notre club dans notre compte-rendu hebdomadaire (nous en sommes actuellement au numéro 23). Mais, si vous le souhaitez bien sûr, nous avons le plaisir de vous annoncer que d'autres sous-programmes en assembleur pour votre Mini-mémoire, baptisés CALL LINL ("LI"), ("SI"), ("CS1ON"), ("CS1OFF") et autres ("TEXTE") ou ("DIM"), seront bientôt à votre disposition.

Alors, lisez 99 Magazine évidemment, dans lequel nous espérons dans un avenir proche vous donner notre solution à quelques problèmes. Par exemple, maintenant que nous sommes capables de créer des dessins, il serait souhaitable de pouvoir les sauvegarder; or, les routines Texas (disquettes ou cassettes) ne conviennent pas pour des raisons telles que :

- un buffer en RAM VDP ne laisse que très peu de place;
- affichage des messages non compatible avec le mode Bit-Map.

Notre cher vieux TI-99/4A, malgré son âge, a encore beaucoup à nous offrir et, avec sa Mini-mémoire chargée avec des sous-programme en assembleur, il sera capable encore longtemps de rivaliser avec les meilleurs.

□

LISSAJOUS

```
10 REM  LISSAJOUS
100 CALL LOAD(-31890,56,1)
110 CALL LINK("BITMAP")
120 YD=186
130 XD=128
140 FOR T=1 TO 360 STEP 4
150 C=16*INT(2+14*RND)
160 CALL LINK("COLOR",C)
170 YA=96+INT(90*COS(T/56))
180 XA=128+INT(110*SIN(T/14))
190 CALL LINK("DROITE",YD,XD,YA,XA)
200 YD=YA
210 XD=XA
220 NEXT T
230 CALL KEY(5,R,E)
240 IF E=0 THEN 230
250 CALL PEEKV(-32768,X)
```

KALEIDOSCOPE

```
10 REM  KALEIDOSCOPE
100 CALL LOAD(-31890,56,0)
110 CALL CLEAR
120 PRINT "            KALEIDOSCOPE":"
    ____________":::
130 PRINT " LES COMMANDES SONT:":::" C
     POUR CHANGER LA"::"          COULEUR
"::
140 PRINT " D,E,S,X POUR DEPLACER LE"::"
        CURSEUR"::" (ENTER) POUR COMMEN
CER ET"::"          FINIR":::
150 INPUT "(ENTER)":R$
160 CALL LINK("BITMAP")
170 Y=20
180 X=100
190 C=32
200 CALL LINK("COLOR",C)
210 CALL LINK("POINT",Y,X)
220 X0=X
230 X=32+Y
240 Y=X0-32
250 CALL LINK("POINT",Y,X)
260 X=256-X
270 CALL LINK("POINT",Y,X)
280 Y=192-Y
290 CALL LINK("POINT",Y,X)
300 X=256-X
310 CALL LINK("POINT",Y,X)
320 X0=X
330 X=32+Y
340 Y=X0-32
350 CALL LINK("POINT",Y,X)
360 Y=192-Y
370 CALL LINK("POINT",Y,X)
380 X=256-X
390 CALL LINK("POINT",Y,X)
400 Y=192-Y
410 CALL KEY(5,R,E)
420 IF E=0 THEN 410
430 IF R<>13 THEN 460
440 CALL PEEKV(-32768,X)
450 END
460 IF R<>67 THEN 490
470 C=C+16+240*(C=240)
480 GOTO 200
490 IF R<>68 THEN 520
500 X=X+1+(X=255)
510 GOTO 210
520 IF R<>83 THEN 550
530 X=X-1-(X=8)
540 GOTO 210
550 IF R<>69 THEN 580
560 Y=Y-1-(Y=1)
570 GOTO 210
580 IF R<>88 THEN 410
590 Y=Y+1+(Y=191)
600 GOTO 210
```

LINES

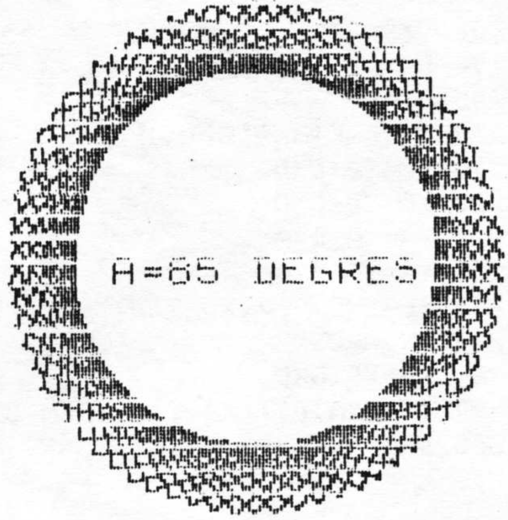
```
10 REM  LINES
100 CALL SCREEN(1)
110 CALL LOAD(-31890,56,0)
120 N=0
130 XD=128
140 YD=96
150 XA=211
160 YA=163
170 DXD=2
180 DYD=4
190 DXA=6
200 DYA=8
210 CALL LINK("BITMAP")
220 XD=XD+DXD
230 YD=YD+DYD
240 XA=XA+DXA
250 YA=YA+DYA
260 IF (XD>0)*(XD<256)THEN 280
270 DXD=-DXD
280 XD=XD+DXD
290 IF (XA>0)*(XA<256)THEN 310
300 DXA=-DXA
310 XA=XA+DXA
320 IF (YD>0)*(YD<192)THEN 340
330 DYD=-DYD
340 YD=YD+DYD
350 IF (YA>0)*(YA<192)THEN 370
360 DYA=-DYA
370 YA=YA+DYA
380 C=16*INT(2+14*RND)
390 CALL LINK("COLOR",C)
400 CALL LINK("DROITE",YD,XD,YA,XA)
410 N=N+1
420 IF N<80 THEN 220
430 FOR I=1 TO 20
440 CALL KEY(5,R,E)
450 IF E=0 THEN 480
460 CALL PEEKV(-32768,X)
470 END
480 NEXT I
490 N=0
500 CALL LINK("BITMAP")
510 DXD=-8+INT(16*RND)
520 DXA=-8+INT(16*RND)
530 DYD=-8+INT(16*RND)
540 DYA=-8+INT(16*RND)
550 GOTO 210
```

POLYGONES

```
10 REM  POLYGONES
100 CALL LOAD(-31890,56,0)
110 CALL CLEAR
120 PRINT "          *POLYGONE*"::::::
130 INPUT "ANGLE ? ":DA
140 YC=96
150 XC=128
160 R=70
170 C=240
180 CALL LINK("BITMAP")
190 YD=YC
200 XD=XC+R
210 P=ATN(1)/45
220 A=A+DA
230 A=A-360*INT(A/360)
240 RD=A*P
250 XA=XC+R*COS(RD)
260 YA=YC+4/3*R*SIN(RD)
270 CALL LINK("DROITE",YD,XD,YA,XA)
280 IF INT(A)=0 THEN 320
290 YD=YA
300 XD=XA
310 GOTO 220
320 CALL SOUND(100,440,0)
330 CALL KEY(5,R,E)
340 IF E=0 THEN 330
350 CALL PEEKV(-32768,X)
```

Routine BITMAP appelée par CALL LINK ("BITMAP")

```
7D00           AORG >7D00
7D00 0002      DATA >0002,>01E0
7D02 01E0
7D04 0206      DATA >0206,>03FF
7D06 03FF
7D08 0403      DATA >0403,>0536
7D0A 0536
7D0C 0603      DATA >0603,>0700
7D0E 0700
7D10 0201      LI   R1,>7D00
7D12 7D00
7D14 0202      LI   R2,>0008
7D16 0008
7D18 C031      MOV  *R1+,R0
7D1A 0420      BLWP @>6034
7D1C 6034
7D1E 0602      DEC  R2
7D20 16FB      JNE  >7D18
7D22 0200      LI   R0,>1800
7D24 1800
7D26 0202      LI   R2,>0003
7D28 0003
7D2A 04C1      CLR  R1
7D2C 0420      BLWP @>6024
7D2E 6024
7D30 0580      INC  R0
7D32 0221      AI   R1,>0100
7D34 0100
7D36 16FA      JNE  >7D2C
7D38 0602      DEC  R2
7D3A 16F7      JNE  >7D2A
7D3C 0200      LI   R0,>1820
7D3E 1820
7D40 04C1      CLR  R1
7D42 0420      BLWP @>6024
7D44 6024
7D46 0220      AI   R0,>0020
7D48 0020
7D4A 0280      CI   R0,>1900
7D4C 1900
7D4E 16F9      JNE  >7D42
7D50 0200      LI   R0,>1877
7D52 1877
7D54 0201      LI   R1,>2000
7D56 2000
7D58 0420      BLWP @>6024
7D5A 6024
7D5C 0580      INC  R0
7D5E 0221      AI   R1,>2000
7D60 2000
7D62 16FA      JNE  >7D58
7D64 0200      LI   R0,>2000
7D66 2000
7D68 0202      LI   R2,>1800
7D6A 1800
7D6C 0420      BLWP @>6024
7D6E 6024
7D70 0580      INC  R0
7D72 0602      DEC  R2
7D74 16FB      JNE  >7D6C
7D76 04C0      CLR  R0
7D78 0202      LI   R2,>03B8
7D7A 03B8
7D7C 0420      BLWP @>6024
7D7E 6024
7D80 0580      INC  R0
7D82 0602      DEC  R2
7D84 16FB      JNE  >7D7C
7D86 0200      LI   R0,>03F0
7D88 03F0
7D8A 0202      LI   R2,>1410
7D8C 1410
7D8E 0420      BLWP @>6024
7D90 6024
7D92 0580      INC  R0
7D94 0602      DEC  R2
7D96 16FB      JNE  >7D8E
7D98 0200      LI   R0,>1B00
7D9A 1B00
7D9C 0201      LI   R1,>D000
7D9E D000
7DA0 0420      BLWP @>6024
7DA2 6024
7DA4 0202      LI   R2,>F000
7DA6 F000
7DA8 045B      B    *R11
```



Routine DROITE appelée par CALL LINK ("DROITE",YD,XD, YA,XA)

```
7E2A C3CB     MOV   R11,R15
7E2C 04C0     CLR   R0
7E2E 0201     LI    R1,>0001
7E30 0001
7E32 0420     BLWP  @>6044
7E34 6044
7E36 0420     BLWP  @>601C
7E38 601C
7E3A 1200     DATA  >1200
7E3C C2A0     MOV   @>834A,R10
7E3E 834A
7E40 0581     INC   R1
7E42 0420     BLWP  @>6044
7E44 6044
7E46 0420     BLWP  @>601C
7E48 601C
7E4A 1200     DATA  >1200
7E4C C260     MOV   @>834A,R9
7E4E 834A
7E50 0581     INC   R1
7E52 0420     BLWP  @>6044
7E54 6044
7E56 0420     BLWP  @>601C
7E58 601C
7E5A 1200     DATA  >1200
7E5C C3A0     MOV   @>834A,R14
7E5E 834A
7E60 0581     INC   R1
7E62 0420     BLWP  @>6044
7E64 6044
7E66 0420     BLWP  @>601C
7E68 601C
7E6A 1200     DATA  >1200
7E6C C360     MOV   @>834A,R13
7E6E 834A
7E70 04CC     CLR   R12
7E72 0205     LI    R5,>0001
7E74 0001
7E76 0206     LI    R6,>0001
7E78 0001
7E7A C1CD     MOV   R13,R7
7E7C 61C9     S     R9,R7
7E7E 1101     JLT   >7E82
7E80 1002     JMP   >7E86
7E82 0505     NEG   R5
7E84 0507     NEG   R7
7E86 C1C7     MOV   R7,R7
7E88 1601     JNE   >7E8C
7E8A 070C     SETO  R12
7E8C C20E     MOV   R14,R8
7E8E 620A     S     R10,R8
7E90 1101     JLT   >7E94
7E92 1002     JMP   >7E98
7E94 0506     NEG   R6
7E96 0508     NEG   R8
7E98 06A0     BL    @>7DCE
7E9A 7DCE
7E9C 8349     C     R9,R13
7E9E 1603     JNE   >7EA6
7EA0 838A     C     R10,R14
7EA2 1601     JNE   >7EA6
7EA4 045F     B     *R15
7EA6 C30C     MOV   R12,R12
7EA8 1103     JLT   >7EB0
7EAA A245     A     R5,R9
7EAC 6308     S     R8,R12
7EAE 10F4     JMP   >7E98
7EB0 A286     A     R6,R10
7EB2 A307     A     R7,R12
7EB4 10F1     JMP   >7E98
7EB6 10EE     JMP   >7E94
```

Routine POINT appelée par CALL LINK ("POINT"),Y,X)

```
7DAA 04C0     CLR   R0
7DAC 0201     LI    R1,>0001
7DAE 0001
7DB0 0420     BLWP  @>6044
7DB2 6044
7DB4 0420     BLWP  @>601C
7DB6 601C
7DB8 1200     DATA  >1200
7DBA C2A0     MOV   @>834A,R10
7DBC 834A
7DBE 0581     INC   R1
7DC0 0420     BLWP  @>6044
7DC2 6044
7DC4 0420     BLWP  @>601C
7DC6 601C
7DC8 1200     DATA  >1200
7DCA C260     MOV   @>834A,R9
7DCC 834A
7DCE 0203     LI    R3,>8000
7DD0 8000
7DD2 0289     CI    R9,>00FF
7DD4 00FF
7DD6 1B28     JH    >7E28
7DD8 0289     CI    R9,>0008
7DDA 0008
7DDC 1A25     JL    >7E28
7DDE 028A     CI    R10,>00BF
7DE0 00BF
7DE2 1B22     JH    >7E28
7DE4 C009     MOV   R9,R0
7DE6 0930     SRL   R0,3
7DE8 C04A     MOV   R10,R1
7DEA 0931     SRL   R1,3
7DEC 0A51     SLA   R1,5
7DEE A001     A     R1,R0
7DF0 0220     AI    R0,>1800
7DF2 1800
7DF4 04C1     CLR   R1
7DF6 0420     BLWP  @>602C
7DF8 602C
7DFA 0951     SRL   R1,5
7DFC C00A     MOV   R10,R0
7DFE 0960     SRL   R0,6
7E00 0AB0     SLA   R0,11
7E02 A040     A     R0,R1
7E04 C009     MOV   R9,R0
7E06 0240     ANDI  R0,>0007
7E08 0007
7E0A 0B03     SRC   R3,0
7E0C C00A     MOV   R10,R0
7E0E 0240     ANDI  R0,>0007
7E10 0007
7E12 A001     A     R1,R0
7E14 0420     BLWP  @>602C
7E16 602C
7E18 E043     SOC   R3,R1
7E1A 0420     BLWP  @>6024
7E1C 6024
7E1E 0220     AI    R0,>2000
7E20 2000
7E22 C042     MOV   R2,R1
7E24 0420     BLWP  @>6024
7E26 6024
7E28 045B     B     *R11
```

Routine COLOR appelée par CALL LINK ("COLOR",C)

```
7EB8 04C0   CLR  R0
7EBA 0201   LI   R1,>0001
7EBC 0001
7EBE 0420   BLWP @>6044
7EC0 6044
7EC2 0420   BLWP @>601C
7EC4 601C
7EC6 1200   JLE  >7EC8
7EC8 C0A0   MOV  @>834A,R2
7ECA 834A
7ECC 06C2   SWPB R2
7ECE 045B   B    *R11
```

Table REF/DEF Le pointeur de début de table est en >701E

```
7FD0        AORG  >7FD0
7FD0 434F   TEXT 'COLOR '
7FD2 4C4F
7FD4 5220
7FD6 7EB8   DATA >7EB8
7FD8 4452   TEXT 'DROITE'
7FDA 4F49
7FDC 5445
7FDE 7E2A   DATA >7E2A
7FE0 504F   TEXT 'POINT '
7FE2 494E
7FE4 5420
7FE6 7DAA   DATA >7DAA
7FE8 4249   TEXT 'BITMAP'
7FEA 544D
7FEC 4150
7FEE 7D10   DATA >7D10
701E        AORG >701E
701E 7FD0   DATA >7FD0
            END
```

Le train sifflera trois fois...

Denise Amrouche, Gilbert Arribet et Jean Sahal

...dès qu'il sera en vue de la bourgade dont vous êtes le chef de gare. Mais les choses se compliquent lorsque deux trains arrivent en même temps et doivent se croiser. En effet, à cet endroit, la gare est traversée par une voie unique et vous disposez d'une voie de garage tout juste assez grande pour contenir la locomotive et un autre wagon, et rien de plus. Pour corser le tout, les chauffeurs de locomotive ne comprennent pas le français... (vous avez compris, c'est parce qu'ils veulent que vous perfectionniez votre anglais !).

Votre mission, si vous l'acceptez, consistera donc à diriger la manœuvre compliquée qui permettra à chaque train de poursuivre sa route. Pour cela, vous envoyez des messages successifs aux chefs de train concernés. Dans la mesure où votre message est rationnel et compréhensible, il est exécuté et vous assistez à l'opération. Dans le cas contraire, rien ne se passe et vous recevez un message d'explication qui apparaît sur votre téléscripteur. Toute erreur est comptabilisée. Si vous provoquez une collision, vous êtes renvoyé. Si vous faites très peu d'erreurs, courez vite vous faire embaucher par les chemins de fer britanniques.

Une recommandation utile : lorsque vous envoyez un message, assurez-vous que vous ne vous êtes pas trompé de destinataire. C'est LEW (avec un "L" comme "Londres") qui conduit la locomotive qui se dirige vers Londres; BOB (avec un "B" comme "Bristol") qui a la charge de la locomotive à destination de Bristol. Pour ceux qui ont des lacunes en géographie, Londres est à l'est, Bristol est à l'ouest.

Du point de vue pédagogique, ce programme permet de comprendre in situ, un point de grammaire anglaise délicat : l'emploi de certains verbes de mouvement, ceci au moyen d'une simulation rendue possible grâce aux nombreuses possibilités graphiques et sonores du TI-99 associé au Basic Etendu. Rien n'y manque : le teuf teuf de la locomotive, le crépitement des téléscripteurs, le fracas éventuel en cas de collision et même le leit-motiv du célèbre film avec Gary Cooper et Grace Kelly.

□

```
100 !      DEVENEZ CHEF DE GARE A LONDRES
110 !
120 !
130 !     PAR DENISE AMROUCHE ET JEAN SAH
AL
140 !
150 !
160 !     TI 99/4A BASIC ETENDU
170 !
180 CALL CHAR(136,"0F1810300007830333337FF
FFF7F3B7BF900B040003F2424243CFFFFFFFFDED
E8C")
190 CALL CHAR(140,"00000000007F7F7F7F7F7
F7FFF9E1E0C0000000000FEFEFEFEFEFEFEFF797
830")
200 CALL CHAR(120,"0609003CE424243FFFFFF
FFFFFFB79303C409E0C0CCCFEFFFFFEFCDEDFC08
000")
210 CALL CHAR(124,"0815123000783033337FF
FFF7F3B79F84080073C2424273FFFFFFFFEDECC8
")
220 CALL CHAR(116,"000000000077F7F7F7F7
F7F7F5E8C80000000003EFEFEFEFFFFFDF8783")
230 CALL CHAR(108,"917139218120CC2C40E2F
A20002DCDC0C3C3C690101010D00101F903973E8
C88")
240 CALL CLEAR :: CALL SCREEN(3):: CALL
MAGNIFY(3)
250 DISPLAY AT(15,1):"  THE STATION MAST
ER GAME"
260 GOSUB 470 :: CALL SCREEN(8)
270 DISPLAY AT(17,1):"YOU ARE THE STATIO
N MASTER":"OF A SMALL AND QUIET RAILWAYS
TATION. BY THE WAY WHAT IS"
280 DISPLAY AT(20,1):"YOUR NAME?" :: ACC
EPT AT(20,12)SIZE(9)VALIDATE(UALPHA)BEEP
:NOM$ :: DISPLAY AT(20,1):" "
290 MM$=NOM$&" - MESSAGE RECIEVED - PRES
S ANY KEY TO START THE MOTION."
300 DISPLAY AT(15,1):"":"SOMETIMES YOU S
EE THE TRAIN COMING FROM BRISTOL ON ITS
 WAY TO LONDON":""
310 GOSUB 1160 :: CALL DEL(400):: GOSUB
1400 :: CALL LHAU :: GOSUB 1200 :: CALL
DELSPRITE(ALL)
320 DISPLAY AT(15,1):"":"":"SOMETIMES YO
U SEE THE LONDONTRAIN GOING TO BRISTOL":
"":""
330 GOSUB 1160 :: CALL DEL(600):: GOSUB
1430 :: CALL BBAS :: GOSUB 1200 :: CALL
DELSPRITE(ALL):: CALL TOUCH
340 DISPLAY AT(15,1):"":"NOTHING REALLY
HAPPENS ...  YET...OCCASIONALLY YOU MAY
 HAVE SOME PROBLEMS":"":"HERE IS ONE EXA
MPLE...":""
350 CALL DEL(200):: GOSUB 1400 :: GOSUB
1430 :: CALL LHAU :: CALL BBAS :: GOSUB
1160
360 GOSUB 1450
370 CALL DEL(150):: CALL BLAN :: DISPLAY
 AT(17,1):"GOOD JOB ";NOM$:"THANKS HEAVE
NS!":"WE HAD A NARROW ESCAPE" :: CALL DE
L(200):: CALL TOUCH
380 GOSUB 1570
390 DISPLAY AT(15,1):"CONGRATULATIONS NO
W YOU ARE READY TO SEND MESSAGES  OF  YO
UR OWN WITHOUT ANY HELP.  WELL GO AHEAD.
"
400 GOSUB 1400 :: GOSUB 1430 :: ER=0 ::
CALL TOUCH
410 FOR MOVE=1 TO 10
420 DISPLAY AT(15,1)SIZE(11):"MOVE NR."&
STR$(MOVE)&":" :: GOSUB 1680 :: ON MOVE
GOSUB 700,750,790,840,880,930,980,1020,1
070,1120
430 NEXT MOVE
440 CALL BLAN :: DISPLAY AT(15,1):NOM$:"
YOU 'VE DONE IT!!..."
450 GOSUB 1290 :: CALL CLEAR :: CALL DEL
SPRITE(ALL):: STOP
460 ! CONFECTION DE LA VOIE
470 CALL COLOR(13,2,4,11,2,4,10,2,4)
480 CALL CHAR(128,"00010FFCE0031FF83EF08
10F7CE00000073EF080000000000000000000000
007")
490 CALL CHAR(112,"00FF0F7CFF031FF83EF08
00FFCE0000000FF0000FF0000070")
500 CALL CHAR(132,"0000000031FF8C0C0000
000000000000FF0000FF00000000FF0000FF1EF
8C")
510 CALL CHAR(104,"00000000011F3D6CCCFFF
FFFFF")
520 CALL HCHAR(1,1,115,416)
530 FOR I=0 TO 9
540 CALL HCHAR(13-I,1+3*I,128):: CALL HC
HAR(13-I,2+3*I,129):: CALL HCHAR(13-I,3+
3*I,130):: CALL HCHAR(13-I,4+3*I,133)
550 CALL HCHAR(12-I,2+3*I,131):: CALL HC
HAR(12-I,3+3*I,132):: NEXT I
560 CALL HCHAR(3,31,128):: CALL HCHAR(3,
32,129):: CALL HCHAR(2,32,131)
570 CALL HCHAR(7,11,104):: CALL HCHAR(8,
11,105)
580 CALL HCHAR(8,12,134,2):: CALL HCHAR(
8,14,114):: CALL HCHAR(8,15,135):: CALL
HCHAR(8,16,112):: CALL HCHAR(8,17,113)
590 RETURN
600 !TRADUCTION DES MESSAGES EN MOUVEMEN
T
610 !0
```

```
620 CALL ORDR(REP$,"LEW","PUSH","TRAIN",
"BACK",TEST,RF$,ER):: IF TEST=0 THEN 640
630 CALL BANDE(24,MM$,1):: CALL LBAS ::
CALL LOCALS(1,75):: CALL ALT :: RETURN
640 GOSUB 1730 :: GOTO 620
650 !00
660 CALL ORDR(REP$,"BOB","PUSH","TRAIN",
"BACK",TEST,RF$,ER):: IF TEST=0 THEN 680
670 CALL BANDE(24,MM$,1):: CALL BHAU ::
CALL LOCALI(4,18):: CALL ALT :: CALL BLA
N :: RETURN
680 GOSUB 1730 :: GOTO 660
690 !1
700 CALL ORDR(REP$,"BOB","PULL","THE YEL
LOW VAN","INTO THE SIDING",TEST,RF$,ER):
: IF TEST=0 THEN 730
710 CALL BANDE(24,MM$,1):: FOR W=4 TO 5
:: CALL BAS(W):: NEXT W :: CALL LOCALS(4
,45):: CALL PATTERN(#4,136):: CALL GAU(4
)
720 CALL LOCALS(5,45):: CALL PATTERN(#5,
140):: CALL GAU(5):: CALL XLOCI(4,96)::
CALL ALT :: CALL BLAN :: RETURN
730 GOSUB 1730 :: GOTO 700
740 !2
750 CALL ORDR(REP$,"LEW","PULL","TRAIN",
"PAST THE SIDING",TEST,RF$,ER):: IF TEST
=0 THEN 770
760 CALL BANDE(24,MM$,1):: CALL LHAU ::
CALL LOCALI(1,35):: CALL ALT :: CALL BLA
N :: RETURN
770 GOSUB 1730 :: GOTO 750
780 !3
790 CALL ORDR(REP$,"LEW","PULL","THE YEL
LOW VAN ","OUT OF THE SIDING",TEST,RF$,E
R):: IF TEST=0 THEN 820
800 CALL BANDE(24,MM$,1):: CALL DTE(5)::
 CALL LHAU :: CALL XLOCS(1,176):: CALL P
ATTERN(#5,116):: CALL HAU(5)
810 CALL LOCALI(1,28):: CALL ALT :: CALL
 BLAN :: RETURN
820 GOSUB 1730 :: GOTO 790
830 !4
840 CALL ORDR(REP$,"LEW","PUSH"," ","BAC
K",TEST,RF$,ER):: IF TEST=0 THEN 860
850 CALL BANDE(24,MM$,1):: CALL BAS(5)::
 CALL LBAS :: CALL LOCALS(1,60):: CALL A
LT :: CALL BLAN :: RETURN
860 GOSUB 1730 :: GOTO 840
870 !5
880 CALL ORDR(REP$,"BOB","MOVE"," ","OUT
 OF THE SIDING",TEST,RF$,ER):: IF TEST=0
 THEN 910
890 CALL BANDE(24,MM$,1):: CALL DTE(4)::
 CALL XLOCS(4,126):: CALL PATTERN(#4,124
):: CALL HAU(4):: CALL LOCALI(4,12):: CA
LL ALT
900 CALL BLAN :: RETURN
910 GOSUB 1730 :: GOTO 880
920 !6
930 CALL ORDR(REP$,"BOB","PULL","THE BLU
E VAN ","INTO THE SIDING",TEST,RF$,ER)::
 IF TEST=0 THEN 960
940 CALL BANDE(24,MM$,1):: CALL BAS(6)::
 CALL BAS(4):: CALL LOCALS(4,45):: CALL
PATTERN(#4,136):: CALL GAU(4):: CALL LOC
ALS(6,45)
950 CALL PATTERN(#6,140):: CALL GAU(6)::
 CALL XLOCI(6,114):: CALL ALT :: CALL BL
AN :: RETURN
960 GOSUB 1730 :: GOTO 930
970 !7
980 CALL ORDR(REP$,"LEW","PULL"," ","PAS
T THE SIDING",TEST,RF$,ER):: IF TEST=0 T
HEN 1000
990 CALL BANDE(24,MM$,1):: CALL LHAU ::
CALL HAU(5):: CALL LOCALI(1,27):: CALL A
LT :: CALL BLAN :: RETURN
1000 GOSUB 1730 :: GOTO 980
1010 !8
1020 CALL ORDR(REP$,"LEW","PULL","THE BL
UE VAN ","OUT OF THE SIDING",TEST,RF$,ER
):: IF TEST=0 THEN 1050
1030 CALL BANDE(24,MM$,1):: CALL DTE(6):
: CALL LHAU :: CALL HAU(5):: CALL XLOCS(
1,199):: CALL PATTERN(#6,116):: CALL HAU
(6)
1040 CALL LOCALI(1,10):: CALL ALT :: CAL
L BLAN :: RETURN
1050 GOSUB 1730 :: GOTO 1020
1060 !9
1070 CALL ORDR(REP$,"BOB","MOVE"," ","OU
T OF THE SIDING",TEST,RF$,ER):: IF TEST=
0 THEN 1100
1080 CALL BANDE(24,MM$,1):: CALL DTE(4):
: CALL XLOCS(4,128):: CALL PATTERN(#4,12
4):: CALL HAU(4):: CALL LOCALI(4,40):: C
ALL ALT
1090 CALL BLAN :: RETURN
1100 GOSUB 1730 :: GOTO 1070
1110 !10
1120 CALL ORDR(REP$,"BOB","PULL"," ","AW
AY",TEST,RF$,ER):: IF TEST=0 THEN 1140
1130 CALL BANDE(24,MM$,1):: CALL BAS(4):
: CALL BAS(6):: CALL BAS(5):: CALL LOCAL
S(4,70):: CALL ALT :: CALL BLAN :: RETUR
N
1140 GOSUB 1730 :: GOTO 1120
1150 ! SIFFLET
```

```
1160 CALL SOUND(100,770,5,800,5,750,5)::
 CALL DEL(50)
1170 CALL SOUND(1500,770,0,800,0,750,0)
1180 RETURN
1190 ! TEUF-TEUF
1200 D=190 :: V=0
1210 FOR I=1 TO 6
1220 IF I=1-3*(I=4)THEN D=D*2
1230 CALL SOUND(D,-5,V,110,V)
1240 IF I=1-3*(I=4)THEN D=D/2
1250 NEXT I
1260 IF D<20 THEN RETURN
1270 D=D/1.25 :: V=V+.5 :: GOTO 1210
1280 !  LE TRAIN SIFFLERA TROIS FOIS
1290 DATA 165,1,220,1,247,1,277,1,220,1,
294,1,277,1,247,1,220,6
1300 DATA 220,1,247,1,277,1,247,1,220,3,
185,4,247,5,185,1,247,1,277,1
1310 DATA 294,1,247,1,330,1,294,1,277,4,
370,4,349,8,330,9
1320 DATA 165,1,220,1,247,1,277,1,220,1,
294,1,277,1,247,1,220,6
1330 DATA 220,1,220,1,247,1,277,1,220,1,
330,1,220,1,370,7,370,1,370,1,415,1,440,
2
1340 DATA 415,1,370,1,330,1,220,6,220,1,
220,1,247,1,277,1,330,4,220,1
1350 DATA 220,1,277,1,330,2,330,1,277,1,
220,2,247,2,220,6
1360 FOR I=1 TO 66 :: READ F1,D
1370 CALL SOUND(D*275,F1,0,F1*1.94,15,F1
*.994,3)
1380 NEXT I :: RETURN
1390 ! CREATION DU TRAIN ALLANT A LONDRE
S
1400 CALL SPRITE(#1,120,2,78,31,#2,116,7
,84,16,#3,116,7,89,1)
1410 RETURN
1420 ! TRAINS VENANT DE LONDRES
1430 CALL SPRITE(#4,124,02,17,213,#5,116
,11,13,229,#6,116,5,8,245):: RETURN
1440 ! TESTER DANGER D'EXPLOSION
1450 CALL BLAN :: CALL SOUND(80,770,5,80
0,5,750,5):: DISPLAY AT(17,1):"QUICK ";M
EC$:"PRESS ANY TOUCH TO AVOID ":"    A
D I S A S T E R "
1460 CALL COINC(#1,#4,15,CC):: CALL KEY(
0,K,S):: IF S<>0 THEN 1490
1470 IF CC=-1 THEN 1480 :: GOTO 1450
1480 CALL EXPLO :: GOTO 1510
1490 CALL ALT :: RETURN
1500 CALL KEY(0,K,S):: IF S=0 THEN 1500
ELSE 1470 :: RETURN
1510 DISPLAY AT(15,1):"W  A  N  T  E  D"
: :"D E A D  OR  A L I V E": :NOM$: :"TH
E STATION MASTER":"WHO  CAUSED A DISASTE
R"
1520 CALL TOUCH :: CALL CLEAR :: CALL DE
LSPRITE(ALL)
1530 PRINT "V A C A N C Y ": : : :"STATI
ON MASTER REQUIRED HIGH": :"SALARY. NOTH
ING SPECIAL TO ": :"DO, JUST WATCH THE T
RAINS": :
1540 PRINT "PASS BY AND OCCASIONALLY USE
": :"A SIDING PROPERLY. PLENTY OF": :"TI
ME FREE TO PLAY WITH YOUR": :
1550 PRINT "PERSONNAL COMPUTER.APPLY TO"
: :"BRITISH RAILWAYS  - LONDON": : : : :
: CALL TOUCH
1560 CALL CLEAR :: CALL SCREEN(3):: REST
ORE :: GOTO 260
1570 DISPLAY AT(15,1):"NOW YOUR WORK BEG
INS.SEND TOEACH ENGINE DRIVER MESSAGES A
BOUT HOW TO USE THE SIDING IN ORDER TO P
ASS. IT IS BIG"
1580 DISPLAY AT(19,1):"ENOUGH TO HOLD ON
E ENGINE   AND ONE WAGON ONLY." :: CALL
TOUCH
1590 DISPLAY AT(15,6):"NOW LET'S SEE IF
YOU CAN":"SEND A MESSAGE CORRECTLY,":"TH
ERE WILL BE SOME WORDS ON THE SCREEN TO
HELP YOU"
1600 CALL TOUCH
1610 DISPLAY AT(1,1)SIZE(27):"INTO/OUT O
F/PAST THE SIDING"
1620 DISPLAY AT(2,1)SIZE(24):"MOVE-PUSH-
PULL-AWAY-BACK"
1630 DISPLAY AT(3,1)SIZE(14):"THE/YOUR T
RAIN"
1640 DISPLAY AT(4,1)SIZE(14):"THE YELLOW
 VAN"
1650 DISPLAY AT(5,1)SIZE(12):"THE BLUE V
AN"
1660 DISPLAY AT(15,1):"TELL LEW TO MOVE
BACK A BIT" :: GOSUB 1690 :: GOSUB 610
1670 DISPLAY AT(15,1):"NOW TELL BOB THE
SAME." :: GOSUB 1690 :: GOSUB 650 :: GOT
O 390
1680 DISPLAY AT(15,12):"TYPE YOUR MESSAG
E"
1690 DISPLAY AT(16,1):"USE MOSTLY THE WO
RDS ON THE SCREEN; 2 LINES AVAILABLE   P
RESS ENTER AFTER EACH LINE NO SPLIT EXPR
ESSIONS:"
1700 DISPLAY AT(21,1):"TO: ":"(CONT.): "
:"____________________________"
1710 ACCEPT AT(21,04)VALIDATE(UALPHA)SIZ
E(-28)BEEP:REP1$
1720 ACCEPT AT(22,09)VALIDATE(UALPHA)SIZ
E(-28)BEEP:REP2$ :: REP$=REP1$&" "&REP2$
```

```
 :: RETURN
1730 CALL BANDE(24,NOM$&RF$,1):: GOSUB 1
710 :: RETURN
1740 !SERIE DE CALL CECI OU CELA
1750 SUB EXPLO :: CALL ALT :: CALL BLAN
:: CALL SOUND(1000,-6,0,200,0):: FOR SC=
1 TO 15 :: FOR T=1 TO 5 :: NEXT T :: CAL
L SCREEN(SC):: NEXT SC
1760 CALL SCREEN(10):: CALL PATTERN(#1,1
08,#2,108,#4,108,#5,108)
1770 XLOC=INT(RND*15+120):: YLOC=INT(RND
*150+1):: CALL LOCATE(#1,YLOC,XLOC)
1780 XLOC=INT(RND*15+150):: YLOC=INT(RND
*150+1):: CALL LOCATE(#4,YLOC,XLOC)
1790 XLOC=INT(RND*15+120):: YLOC=INT(RND
*150+1):: CALL LOCATE(#2,YLOC,XLOC)
1800 XLOC=INT(RND*20+100):: YLOC=INT(RND
*150+10):: CALL LOCATE(#5,YLOC,XLOC)
1810 SUBEND
1820 SUB TOUCH
1830 DISPLAY AT(24,1):"PRESS ANY KEY TO
GO ON"
1840 CALL KEY(0,K,S):: IF S<>1 THEN 1840
1850 CALL BLAN
1860 SUBEND
1870 !MOUVEMENT DES TRAINS
1880 SUB BAS(W)!VERS BRISTOL
1890 CALL MOTION(#W,2,-6):: SUBEND
1900 SUB LBAS !TR VERS LDRES DESCEND
1910 FOR W=3 TO 1 STEP -1 :: CALL MOTION
(#W,2,-6):: NEXT W :: SUBEND
1920 SUB BBAS !TRAIN VERS BRISTOL DESCEN
D
1930 FOR W=4 TO 6 :: CALL MOTION(#W,2,-6
):: NEXT W :: SUBEND
1940 !
1950 SUB HAU(W)!VERS LONDRES
1960 CALL MOTION(#W,-2,6):: SUBEND
1970 SUB LHAU !TR DE LONDRES MONTE
1980 FOR W=1 TO 3 :: CALL MOTION(#W,-2,6
):: NEXT W :: SUBEND
1990 SUB BHAU !TR DE BRISTOL MONTE
2000 FOR W=4 TO 6 :: CALL MOTION(#W,-2,6
):: NEXT W :: SUBEND
2010 !
2020 SUB DTE(W)!SORTIE DE SD
2030 CALL MOTION(#W,0,6):: SUBEND
2040 SUB GAU(W)!ENTRE DS SD
2050 CALL MOTION(#W,0,-6):: SUBEND
2060 SUB STO(W)!W=NUMERO DU SPRITE
2070 CALL MOTION(#W,0,0):: SUBEND
2080 SUB ALT !ARRET TOTAL
2090 FOR W=1 TO 6 :: CALL MOTION(#W,0,0)
:: NEXT W :: SUBEND
2100 SUB BLAN :: DISPLAY AT(14,1):"": :
: : : : : : : : : : :: SUBEND
2110 SUB DEL(D):: FOR T=1 TO D :: NEXT T
 :: SUBEND
2120 SUB BANDE(LI,M$,VIT)
2130 BA$=RPT$(" ",28)&M$
2140 FOR I=1 TO LEN(BA$):: DISPLAY AT(LI
,1):SEG$(BA$,I,28):: CALL SOUND(VIT,4000
0,0):: CALL KEY(0,K,S):: IF S<>0 THEN GO
SUB 1160 :: GOTO 2160
2150 NEXT I :: GOTO 2140
2160 DISPLAY AT(24,1):"" :: SUBEND
2170 SUB LOCALS(W,XX)
2180 CALL POSITION(#W,X,Y):: IF X>=XX TH
EN 2190 ELSE 2180
2190 SUBEND
2200 SUB LOCALI(W,XX)
2210 CALL POSITION(#W,X,Y):: IF X<=XX TH
EN 2220 ELSE 2210
2220 SUBEND
2230 SUB XLOCI(W,XX)
2240 CALL POSITION(#W,X,Y):: IF Y<=XX TH
EN 2250 ELSE 2240
2250 SUBEND
2260 SUB XLOCS(W,XX)
2270 CALL POSITION(#W,X,Y):: IF Y>=XX TH
EN 2280 ELSE 2270
2280 SUBEND
2290 SUB ORDR(REP$,DR$,VE$,CPL$,ADV$,TES
T,RF$,ER)
2300 DR=POS(REP$,DR$,1)
2310 VE=POS(REP$,VE$,1)
2320 CPL=POS(REP$,CPL$,1)
2330 ADV=POS(REP$,ADV$,1)
2340 TEST=DR*VE*CPL*ADV :: IF TEST=0 THE
N ER=ER+1
2350 ERR$="BY THE WAY,YOU MADE "&STR$(ER
)&" MISTAKE"&CHR$(-83*(ER>1))&"."
2360 RF$=" USE THE ABOVE EXPRESSIONS UNS
PLIT MAINLY: "&ADV$&" - "&VE$&" - "&" DR
IVER'S NAME ."&ERR$&"- PRESS ANY KEY TO
GET ANOTHER CHANCE - "
2370 SUBEND
```

Comptes familiaux

Jean-Philippe Guillemant

Présentation

Ce programme de comptes familiaux offre la possibilité de gérer avec le TI, sur disquette, mois par mois et rubrique par rubrique, les recettes et dépenses de la famille, et cela pendant 17 années non obligatoirement consécutives.

A partir des entrées effectuées, le TI pourra afficher l'état des comptes, les bilans mensuels ou annuels, l'évolution des recettes et dépenses sur une année ou l'évolution annuelle d'une rubrique choisie parmi les 18 disponibles en recettes ou les 18 disponibles en dépenses.

Configuration

Il est nécessaire de disposer, avec le TI-99/4A, du module "**Basic Etendu**", de l'extension de mémoire et d'un lecteur de disquettes. Une imprimante serait bienvenue, mais la routine correspondante n'est pas développée (voir paragraphe "Impression des résultats").

Utilisation

Appel

Une disquette complète est réservée à la gestion des comptes, ce qui permet de nommer le programme "DSK1.LOAD" de manière à ce qu'il soit chargé automatiquement à la mise en route du Basic étendu.

Menu principal

Après une phase d'initialisation, pendant laquelle les variables sont préparées et la structure des comptes lue en mémoire, le menu principal est affiché. 7 options sont proposées; il en existe une huitième qui n'est pas affichée car elle est dangereuse : il s'agit de l'option 0 (voir paragraphe suivant). On choisit une option en tapant le numéro correspondant suivi par <ENTER>.

A la fin de chaque fonction, le programme revient au menu principal.

Quand on revient dans une fonction déjà exécutée, y compris dans le menu principal, les réponses tapées précédemment sont affichées à nouveau. Il suffit de taper <ENTER> pour les prendre en compte à nouveau.

Initialisation (option 0)

ATTENTION : cette option est dangereuse; c'est pourquoi elle n'apparaît pas dans le menu. Elle est essentiellement utile en période de test. Elle permet de réinitialiser les fichiers utilisés par le programme. Toutes les informations précédemment saisies sont perdues dès que l'on a répondu "O" à la demande de confirmation.

Entrée des rubriques (option 1)

On dispose de 18 rubriques en recettes et 18 en dépenses. Chaque rubrique a un nom de 18 caractères et une abréviation de 4 caractères (utilisée dans les bilans, voir paragraphe "Bilan comptable").

Le programme affiche les 10 premiers caractères de chaque rubrique. Les noms des rubriques non définies sont remplacées par des points. Chaque rubrique est numérotée.
Il demande ensuite le type de rubrique à créer ou à modifier ("1" pour recettes ou "2" pour dépenses), puis le numéro de la rubrique concernée (1 à 18). Si la rubrique est déjà définie, il affiche son nom et son abréviation. On peut alors entrer un nouveau nom (sur 18 caractères maximum) et une nouvelle abréviation (sur 4 caractères maximum). Si on ne rentre aucun caractères pour le nom, la rubrique est annulée et l'abréviation n'est pas demandée.

On peut modifier plusieurs rubriques successivement en répondant "O" à la question "ENCORE (O/N)". Quand on répond "N", le fichier des rubriques est mis à jour et le programme revient au menu principal.

Les options suivantes (options 2 à 6) ne peuvent pas être appelées tant que le fichier des rubriques n'est pas créé.

Entrée des comptes (option 2)

Le programme commence par demander l'année (sur 2 chiffres), le mois (de 1 à 12) et le type de rubriques ("1" pour Recettes ou "2" pour Dépenses) concernés. Après avoir chargé en mémoire les comptes correspondants (s'ils existent), il demande si on désire modifier le montant d'une rubrique ou de toutes.

Si on répond "O" à cette question, il demande la rubrique concernée après avoir affiché, pour mémoire, le numéro et le nom (les 10 premiers caractères) des rubriques. On peut alors mettre à jour cette rubrique. Le nouveau montant est affiché immédiatement. Quand on ne veut pas modifier le montant, il suffit de taper <ENTER>.

Ces modifications peuvent être effectuées tant que l'on répond "O" à la question "ENCORE (O/N)". On enregistre les modifications et on revient au menu principal en répondant "N" à cette question.

Si on a répondu "N" à la question "MODIFICATION D'UNE RUBRIQUE", le programme affiche toutes les rubriques pour ce mois et ce type et met le curseur en face du premier montant. On peut le laisser tel quel en tapant <ENTER> ou le modifier. Quand toutes les rubriques ont été passées en revue, si on répond "O" à la question "CORRECT (O/N)", les nouveaux comptes sont enregistrés et on revient au menu principal, sinon on recommence la vérification.

Etat des comptes (option 3)

Cette fonction indique, sous forme de tableau par année et par mois, les comptes déjà saisis. La lettre "R" est affichée si les recettes existent, la lettre "D" si les dépenses existent, le symbole "RD" si recettes et dépenses existent ou un tiret si les comptes correspondants n'ont pas été créés.

On revient au menu principal en appuyant sur la touche <ENTER> (voir paragraphe "Impression des résultats").

Bilan comptable (option 4)

La fonction demande si on souhaite obtenir le bilan correspondant à un mois donné ou à une année complète. Il faut répondre "1" pour un bilan mensuel (réponse par défaut) ou "2" pour un bilan annuel. Elle demande ensuite l'année désirée et, éventuellement, le mois.

Après avoir fait les calculs correspondants, elle affiche l'ensemble des recettes et dépenses pour le mois ou l'année concerné, ainsi que les totaux et le solde. C'est dans cette fonction que sont utilisées les abréviations des noms de rubriques. On revient au menu principal en appuyant sur <ENTER> (voir paragraphe "Impression des résultats").

Evolution annuelle (option 5)

Cette fonction permet de visualiser l'évolution, mois par mois, des recettes et des dépenses d'une année donnée (même si les comptes de l'année ne sont pas complets).

Après avoir demandé l'année concernée, le système affiche un tableau indiquant le total des recettes et le total des dépenses de chaque mois, ainsi que les totaux et le solde annuels.

Après que l'on ait appuyé sur <ENTER> (voir paragraphe "Impression des résultats"), le système affiche les mêmes résultats, mais sous forme d'un graphique en bâtons verticaux, les recettes vers le haut de l'écran et les dépenses vers le bas. Le graphique est gradué en dixièmes du montant le plus élevé et la valeur de l'unité de graduation est indiquée sous le graphique. On revient au menu principal en appuyant sur <ENTER> (voir paragraphe "Impression des résultats").

Evolution d'une rubrique (option 6)

Il est possible, grâce à cette fonction, de connaître l'évolution, pour une année donnée, d'une rubrique quelconque. Le programme demande tout d'abord le type de rubrique ("1" pour recettes ou "2" pour dépenses) et l'année désirée. Après avoir affiché, pour mémoire, les numéros et libellés des rubriques (sur 10 caractères), il demande le numéro de la rubrique que l'on souhaite voir évoluer.

Le système affiche alors, mois par mois, les montants associés à la rubrique choisie, ainsi que la moyenne mensuelle et le total annuel.

Quand on appuie sur <ENTER> (voir paragraphe "Impression des résultats"), le système montre l'évolution de la rubrique sous la forme d'un graphique en bâtons horizontaux. Ce graphique est gradué en vingtièmes du montant le plus élevé et la valeur de l'unité de graduation est indiquée sous le graphique. On revient au menu principal en tapant <ENTER> (voir paragraphe "Impression des résultats").

Sortie du programme (option 7)

Cette fonction permet bien évidemment de terminer la gestion des comptes. Il est obligatoire de sortir par cette option car l'action des touches <BREAK> et <QUIT> est interdite.

Impression des résultats

Ce programme est particulièrement intéressant si on dispose d'une imprimante pour conserver les tableaux. A défaut d'en avoir, seul l'appel de l'impression est actuellement prévu.

Cela doit se faire par appui sur les touches "P" et <ENTER> après l'affichage des tableaux (options 3, 4, 5 et 6). La routine correspondante s'appelle "IMP_ECR" et n'a qu'à effectuer une recopie d'écran.

Erreurs

En cas d'erreur pendant l'exécution, le programme affiche le code d'erreur (cf. l'appendice N du manuel Basic Etendu) et le numéro de ligne et attend que l'on ait appuyé sur <ENTER> pour se terminer.

Le programme

Structure du programme

Le programme est organisé en modules distincts pour chaque fonction, respectivement :

INITIALISATION	lignes	100 - 530
MENU PRINCIPAL	lignes	540 - 720
ERREURS	lignes	730 - 760
OPTION 0	lignes	770 - 850
OPTION 1	lignes	860 - 1020
OPTION 2	lignes	1030 - 1550
OPTION 3	lignes	1560 - 1680
OPTION 4	lignes	1690 - 2000
OPTION 5	lignes	2010 - 2360
OPTION 6	lignes	2370 - 2750
IMPRESSION	lignes	2760 - 2810

chacun de ces modules étant indépendant des autres pour des facilités de programmation et de corrections.

Description des fichiers

Fichier des rubriques

Le nom de ce fichier est rangé dans dans la variable F_RUB$. Il est défini en enregistrements de longueur variables de 48 octets chacun. Il contient 18 enregistrements. La structure de ces enregistrements est la suivante :

- libellé recette 19 octets
- abréviation recette 5 octets
- libellé dépense 19 octets
- abréviation dépense 5 octets

pour chacune des 18 rubriques possibles.

Fichiers des comptes

Le nom de ce fichier est rangé dans la variable F_MONT$. Il est défini en enregistrements de de longueur fixe de manière à pouvoir y accéder en accès direct. La longueur des

enregistrements est de 162 octets, soit 18 nombres en représentation interne (9 octets chacun). Le contenu des enregistrements est différents selon leur position :

- Enregistrement 0

Il permet de repérer les années existantes. Les 17 premiers nombres contiennent soit 0, soit un numéro d'année créée. Si l'année existe, sa position dans cet enregistrement permet de pointer sur les enregistrements concernés (voir "Enregistrements 1 à 34"). Ceci permet de créer les années dans le désordre.

Le dernier nombre donne le nombre d'enregistrements existant dans le fichier, de manière à pouvoir en ajouter.

- Enregistrements 1 à 34

Il existe deux enregistrements de ce type par année. Le premier pointe sur les recettes de l'année, le deuxième sur les dépenses. Seuls les 12 premiers nombres sont utilisés. Ils contiennent respectivement pour chaque mois de l'année soit 0, soit le numéro de l'enregistrement des comptes correspondants si ceux-ci existent. Ceci permet de créer les comptes mensuels dans le désordre.

- Enregistrements 35 et suivants

Ils contiennent les montants saisis, soit en recettes, soit en dépenses, dans l'ordre des numéros de rubriques.

Redéfinition des caractères

Code 91 : dessin diagramme option 5, recettes.
Code 92 : double souligné.
Code 93 : dessin diagramme option 5, dépenses.
Code 94 : dessin diagramme option 6.
Code 95 : souligné et cadre tableaux.
Code 124 : cadre tableaux.
Code 126 : symboles recettes/dépenses, option 3.
Codes 128 à 135 : bâtons recettes option 5, bâtons option 6.
Codes 136 à 143 : bâtons dépenses option 5.

Variables

Toutes les variables sont réservées au début du programme.

Variables générales

Elles sont utilisées dans l'ensemble du programme.

- **FRUB** Vrai (-1) si le fichier rubriques existe. Faux (0) sinon (valeur initiale).
- **RECN** Nombre d'enregistrements du fichier comptes (vaut 35 à l'initialisation).
- **OFF_AN** Déplacement de l'année dans l'enregistrement 0 du fichier comptes; permet d'accéder aux enregistrements recettes et dépenses.
- **F_RUB$** Contient le nom du fichier des rubriques.
- **F_MONT$** Contient le nom du fichier des comptes.

A$, I, J et K sont des variables temporaires.

Variables locales

Pour plus de clarté, les variables suivantes sont affectées à chaque module en ajoutant le numéro du module au nom de la variable; cette technique permet également d'afficher la valeur précédente quand on revient dans une fonction :

CHXO : option choisie dans le menu principal.
NR1, NR2, NR6 : numéro de rubrique.
RD1, RD2, RD6 : type de rubrique ("1" pour recettes, "2" pour dépenses).
MA4 : indicateur de traitement mensuel ("1") ou annuel ("2").
ANN2, ANN3, ANN4, ANN5, ANN6 : année en traitement.
MOIS2, MOIS4 : mois en traitement.

Matrices

Les matrices sont initialisées avec "OPTION BASE 1".

AN(17) : années existantes (copie de l'enregistrement 0 du fichier des comptes).
Indice : pointeur sur les enregistrements recettes et dépenses (1 à 34) correspondant à l'année (OFF_AN).

MO(2,12) : pointeurs sur les enregistrements de comptes pour une année donnée (copie de 2 enregistrements successifs 1 à 34 du fichier des comptes).
1er indice : type de rubrique (RD?).
2ème indice : mois (MOIS?).

NOM_M$(12) : noms des mois de l'année.
Indice : numéro du mois (MOIS?).

RUBR$(2,18,2) : copie du fichier des rubriques.

1er indice : type de rubrique (RN?).
2ème indice : numéro de rubrique (NR?).
3ème indice : nom complet (1) ou abréviation (2).

TOTAL(2) : totalisation des comptes.
Indice : type de rubrique (RD?).

MONT(13,2,19) : contient les montants en cours de traitement.
1er indice : mois (MOIS?). Si = 13, il s'agit d'une totalisation annuelle.
2ème indice : type de rubrique (RD?).
3ème indice : numéro de rubrique (NR?). Si = 19, il s'agit d'une totalisation par rubrique.

□

```
100 !====================
110 != COMPTES FAMILIAUX =
120 !====================
130 !
140 ! (c) J.-Ph.Guillemant
150 !
160 OPTION BASE 1 :: CALL CLEAR :: ON BR
EAK NEXT :: CALL INIT :: CALL LOAD(-3180
6,16)!interdire touches 'clear' & 'quit'
165 CALL LOAD(8196,62,248):: CALL LOAD(1
6376,77,73,82,69,32,32,160,0):: CALL LOA
D(-24576,3,0,0,2,2,224,131,224,4,32,0,0)
170 IMAGE ## ##########
180 IMAGE ####.##
190 IMAGE ######.##
200 IMAGE #######.##
210 IMAGE ####|#######.##|#######.##|
220 IMAGE #########:#######.##
230 IMAGE REC#######.##|DEP#######.##|
240 IMAGE 1 U.=#####.##
250 !matrices & variables
260 DIM RUBR$(2,18,2),AN(17),MO(2,12),NO
```

```
M_M$(12),MONT(13,2,19),TOTAL(2)
270 FRUB,CHX0,NR1,NR2,ANN2,MOIS2,ANN3,AN
N4,MOIS4,ANN5,NR6,ANN6=0 :: RD1,RD2,MA4,
RD6=1 :: RECN,OFF_AN,I,J,K=0
280 A$="" :: F_RUB$="DSK1.RUBRIQUES" ::
F_MONT$="DSK1.MONTANTS"
290 !noms des mois
300 DATA Janvier,Fevrier,Mars,Avril,Mai,
Juin,Juillet,Aout,Septembre,Octobre,Nove
mbre,Decembre
310 !batons recettes
320 DATA 00000000000007E0000000000007E7
E00000000007E7E7E000000007E7E7E7E
330 DATA 0000007E7E7E7E7E00007E7E7E7E7E7
E007E7E7E7E7E7E7E7E7E7E7E7E7E7E7E
340 !batons depenses
350 DATA 7E00000000000007E7E00000000000
07E7E7E00000000007E7E7E7E00000000
360 DATA 7E7E7E7E7E0000007E7E7E7E7E7E000
07E7E7E7E7E7E7E007E7E7E7E7E7E7E7E
370 !batons rubriques
380 DATA 00808080808080000000C0C0C0C0C0C00
000E0E0E0E0E0E00000F0F0F0F0F0F000
390 DATA 00F8F8F8F8F8F80000FCFCFCFCFCFC0
000FEFEFEFEFEFE0000FFFFFFFFFFFF00
400 !caracteres: 91([), 92(\), 93(]), 94
(^), 95(_), 124(|), 126(~)
410 CALL CHAR(91,"101010101010107C",92,"
000000FF00FF0000",93,"7C10101010101010",
94,"00008080FF808000")
420 CALL CHAR(95,"00000000FF000000",124,
RPT$("10",8),126,"E0909EE9C9A9090E")
430 ! mise a jour table mois
440 FOR I=1 TO 12 :: READ NOM_M$(I):: NE
XT I
450 ! lecture rubriques
460 ON ERROR 490 :: OPEN #1:F_RUB$,INPUT
 ,INTERNAL,VARIABLE 2*(19+5)
470 ON ERROR 760 :: FOR I=1 TO 18 :: INP
UT #1:RUBR$(1,I,1),RUBR$(1,I,2),RUBR$(2,
I,1),RUBR$(2,I,2):: NEXT I :: CLOSE #1 :
: FRUB=-1
480 ! chargement annees
490 ON ERROR 510 :: OPEN #1:F_MONT$,INPU
T ,INTERNAL,FIXED 9*18
500 ON ERROR 760 :: FOR I=1 TO 17 :: INP
UT #1:AN(I),:: NEXT I :: INPUT #1:RECN :
: CLOSE #1 :: GOTO 570
510 ON ERROR 760 :: OPEN #1:F_MONT$,OUTP
UT,RELATIVE,INTERNAL,FIXED 9*18
520 RECN=35 :: FOR I=1 TO 17 :: PRINT #1
:AN(I),:: NEXT I :: PRINT #1:RECN
530 FOR I=1 TO 17 :: FOR J=1 TO 2 :: FOR
 K=1 TO 11 :: PRINT #1:MO(J,K),:: NEXT K
 :: PRINT #1:MO(J,K):: NEXT J :: NEXT I
:: CLOSE #1
540 !
550 ! menu principal
560 !
570 DISPLAY AT(1,5)ERASE ALL:"COMPTES FA
MILIAUX":TAB(5);RPT$("\",17)
580 RESTORE 640 :: READ I :: FOR J=4 TO
I*2+2 STEP 2 :: READ A$ :: DISPLAY AT(J,
1):A$ :: NEXT J
590 DISPLAY AT(23,1):"QUEL CHOIX DESIREZ
-VOUS?" :: IF CHX0<>0 THEN DISPLAY AT(23
,25):CHX0
600 ACCEPT AT(23,26)VALIDATE(DIGIT)SIZE(
-1)BEEP:A$ :: IF A$="" THEN 600 ELSE CHX
0=VAL(A$):: IF CHX>I THEN 600
610 IF CHX0<>0 AND CHX0<>1 AND CHX0<>I A
ND NOT FRUB THEN DISPLAY AT(24,1)BEEP:"P
AS DE FICHIER RUBRIQUES!!!" :: GOTO 600
620 IF CHX0<>I THEN ON CHX0+1 GOSUB 800,
890,1060,1590,1720,2040,2400 :: GOTO 570
 ELSE CALL LINK("MIRE")
630 !menu principal
640 DATA 7
650 !     0 initialisation fichiers
660 DATA "1 ENTREE DES RUBRIQUES"
670 DATA "2 ENTREE DES COMPTES"
680 DATA "3 ETAT DES COMPTES"
690 DATA "4 BILAN COMPTABLE"
700 DATA "5 EVOLUTION ANNUELLE"
710 DATA "6 EVOLUTION D'UNE RUBRIQUE"
720 DATA "7 FIN DU PROGRAMME"
730 !
740 ! traitement d'erreurs
750 !
760 CALL ERR(I,J,J,J):: DISPLAY AT(24,1)
BEEP:"ERREUR";I;"A LA LIGNE";J :: ACCEPT
 AT(24,28)SIZE(1)BEEP:A$ :: CALL CLEAR :
: STOP
770 !
780 ! choix 0: re-init fichiers
790 !
800 DISPLAY AT(24,1):"ETES-VOUS VRAIMENT
 SUR ? N" :: ACCEPT AT(24,26)SIZE(-1)BEE
P:A$ :: IF A$<>"O" THEN 850
810 DELETE F_RUB$ :: DELETE F_MONT$ :: F
RUB=0 :: FOR I=1 TO 2 :: FOR J=1 TO 18 :
: FOR K=1 TO 2 :: RUBR$(I,J,K)="" :: NEX
T K :: NEXT J :: NEXT I
820 OPEN #1:F_MONT$,OUTPUT,RELATIVE,INTE
RNAL,FIXED 9*18 :: RECN=35 :: FOR I=1 TO
 17 :: AN(I)=0 :: PRINT #1:AN(I),:: NEXT
 I :: PRINT #1:RECN
830 FOR I=1 TO 17 :: FOR J=1 TO 2 :: FOR
 K=1 TO 11 :: MO(J,K)=0 :: PRINT #1:MO(J
,K),:: NEXT K
840 MO(J,K)=0 :: PRINT #1,MO(J,K):: NEXT
 J :: NEXT I :: CLOSE #1
850 RETURN
860 !
870 ! choix 1: modification rubriques
880 !
890 DISPLAY AT(1,4)ERASE ALL:"RECETTES";
TAB(18);"DEPENSES":TAB(4);RPT$("_",8);TA
```

```
B(18);RPT$("_",8)
900 FOR I=1 TO 18 :: FOR J=1 TO 2 :: IF
RUBR$(J,I,1)="" THEN A$=RPT$(".",10)ELSE
 A$=SEG$(RUBR$(J,I,1),1,10)
910 DISPLAY AT(I+3,(J-1)*14+1):USING 170
:I,A$ :: NEXT J :: NEXT I
920 !modification
930 DISPLAY AT(23,1):"ENTREZ 1 POUR RECE
TTE":TAB(8);"2 POUR DEPENSE:";RD1 :: ACC
EPT AT(24,24)VALIDATE("12")SIZE(-1)BEEP:
RD1
940 DISPLAY AT(23,1):"RUBRIQUE A MODIFIE
R:":"" :: IF NR1<>0 THEN DISPLAY AT(23,2
1):NR1
950 ACCEPT AT(23,22)VALIDATE(DIGIT)SIZE(
-2)BEEP:A$ :: IF A$="" THEN 950 ELSE NR1
=VAL(A$):: IF NR1<1 OR NR1>18 THEN 950
960 DISPLAY AT(23,1):"ENTREZ LE NOM DE L
A RUBRIQUE":RUBR$(RD1,NR1,1);TAB(22);RUB
R$(RD1,NR1,2):: ACCEPT AT(24,1)SIZE(-18)
BEEP:RUBR$(RD1,NR1,1)
970 DISPLAY AT(NR1+3,(RD1-1)*14+4)SIZE(1
0):RUBR$(RD1,NR1,1):: IF RUBR$(RD1,NR1,1
)="" THEN RUBR$(RD1,NR1,2)="" :: GOTO 99
0
980 DISPLAY AT(23,1):"ABREVIATION (4 CAR
.) ?" :: DISPLAY AT(24,22)SIZE(4):RUBR$(
RD1,NR1,1):: ACCEPT AT(24,22)SIZE(-4)BEE
P:RUBR$(RD1,NR1,2)
990 DISPLAY AT(23,1):"ENCORE (O/N): O":"
" :: ACCEPT AT(23,15)VALIDATE("ON")SIZE(
-1)BEEP:A$ :: IF A$="O" THEN 930
1000 !ecriture disque
1010 OPEN #1:F_RUB$,OUTPUT,INTERNAL,VARI
ABLE 2*(19+5)
1020 FOR I=1 TO 18 :: PRINT #1:RUBR$(1,I
,1),RUBR$(1,I,2),RUBR$(2,I,1),RUBR$(2,I,
2):: NEXT I :: CLOSE #1 :: FRUB=-1 :: RE
TURN
1030 !
1040 ! choix 2: entree des comptes
1050 !
1060 DISPLAY AT(1,5)ERASE ALL:"ENTREE DE
S COMPTES":TAB(5);RPT$("\",18): :"ANNEE
(19XX):": :"MOIS  (1-12):"
1070 DISPLAY AT(8,1):"RECETTES (1)":"DEP
ENSES (2):";RD2 :: IF ANN2<>0 THEN DISPL
AY AT(4,14):ANN2
1080 IF MOIS2<>0 THEN DISPLAY AT(6,14):M
OIS2
1090 ACCEPT AT(4,15)VALIDATE(DIGIT)SIZE(
-2)BEEP:A$ :: IF A$="" THEN 1090 ELSE AN
N2=VAL(A$)
1100 ACCEPT AT(6,15)VALIDATE(DIGIT)SIZE(
-2)BEEP:A$ :: IF A$="" THEN 1100 ELSE MO
IS2=VAL(A$):: IF MOIS2<1 OR MOIS2>12 THE
N 1100
1110 ACCEPT AT(9,15)VALIDATE("12")SIZE(-
1)BEEP:RD2
```

```
1120 !recherche montants par type & mois
1130 OFF_AN=0 :: J=1 :: FOR I=1 TO 17 ::
 IF AN(I)=ANN2 THEN OFF_AN=I*2-1 ELSE IF
 AN(I)<>0 THEN J=J+1
1140 NEXT I :: IF OFF_AN<>0 THEN 1160
1150 IF J=18 THEN DISPLAY AT(24,2)BEEP:"
PAS PLUS DE 17 ANNEES !!" :: GOTO 1090 E
LSE OFF_AN=J*2-1 :: AN(J)=ANN2
1160 DISPLAY AT(24,1):"" :: OPEN #1:F_MO
NT$,INTERNAL,RELATIVE,UPDATE,FIXED 9*18
:: RESTORE #1,REC OFF_AN+RD2-1
1170 FOR I=1 TO 11 :: INPUT #1:MO(RD2,I)
,:: NEXT I :: INPUT #1:MO(RD2,I)
1180 FOR I=1 TO 18 :: MONT(MOIS2,RD2,I)=
0 :: NEXT I :: IF MO(RD2,MOIS2)=0 THEN 1
200
1190 RESTORE #1,REC MO(RD2,MOIS2):: FOR
I=1 TO 17 :: INPUT #1:MONT(MOIS2,RD2,I),
:: NEXT I :: INPUT #1:MONT(MOIS2,RD2,I)
1200 DISPLAY AT(11,1):"MODIF. D'UNE RUBR
IQUE: O" :: ACCEPT AT(11,24)VALIDATE("ON
")SIZE(-1)BEEP:A$ :: IF A$="N" THEN 1380
1210 !modification d'une rubrique
1220 CALL HCHAR(4,1,32,8*32):: DISPLAY A
T(15,1):RPT$("_",28):: FOR I=1 TO 18 ::
IF RUBR$(RD2,I,1)="" THEN 1240
1230 A$=SEG$(RUBR$(RD2,I,1),1,10):: J=IN
T((I-1)/2):: DISPLAY AT(J+16,(I-1-J*2)*1
4+1):USING 170:I,A$
1240 NEXT I
1250 DISPLAY AT(4,1):"NO DE RUBRIQUE :":
"" :: IF NR2<>0 THEN DISPLAY AT(4,17):NR
2
1260 DISPLAY AT(7,1):"ANCIENNE VALEUR:":
"    A AJOUTER...:":"    A DEDUIRE...:":TA
B(18);RPT$("_",8):"NOUVELLE VALEUR:"
1270 ACCEPT AT(4,18)VALIDATE(DIGIT)SIZE(
-2)BEEP:A$ :: IF A$="" THEN 1270 ELSE NR
2=VAL(A$):: IF NR2<1 OR NR2>18 THEN 1270
1280 IF RUBR$(RD2,NR2,1)="" THEN 1270 EL
SE DISPLAY AT(4,1):""
1290 I=LEN(RUBR$(RD2,NR2,1)):: DISPLAY A
T(4,14-I/2):RUBR$(RD2,NR2,1):TAB(14-I/2)
;RPT$("_",I)
1300 DISPLAY AT(7,17):USING 190:MONT(MOI
S2,RD2,NR2):: DISPLAY AT(11,17):USING 19
0:MONT(MOIS2,RD2,NR2)
1310 ACCEPT AT(8,18)VALIDATE(NUMERIC)SIZ
E(-9)BEEP:A$ :: IF A$="" THEN 1330 ELSE
DISPLAY AT(8,17):USING 190:VAL(A$)
1320 MONT(MOIS2,RD2,NR2)=MONT(MOIS2,RD2,
NR2)+VAL(A$):: DISPLAY AT(11,17):USING 1
90:MONT(MOIS2,RD2,NR2)
1330 ACCEPT AT(9,18)VALIDATE(NUMERIC)SIZ
E(-9)BEEP:A$ :: IF A$="" THEN 1350 ELSE
DISPLAY AT(9,17):USING 190:VAL(A$)
1340 MONT(MOIS2,RD2,NR2)=MONT(MOIS2,RD2,
NR2)-VAL(A$):: DISPLAY AT(11,17):USING 1
90:MONT(MOIS2,RD2,NR2)
```

```
1350 DISPLAY AT(13,1):"ENCORE (O/N): O"
:: ACCEPT AT(13,15)VALIDATE("ON")SIZE(-1
)BEEP:A$
1360 IF A$="O" THEN DISPLAY AT(13,1):""
:: GOTO 1250 ELSE 1520
1370 !affichage tous montants
1380 TOTAL(RD2)=0 :: IF RD2=1 THEN A$="R
ECETTES " ELSE A$="DEPENSES "
1390 A$=A$&NOM_M$(MOIS2)&" 19"&STR$(ANN2
):: DISPLAY AT(1,(28-LEN(A$))/2)ERASE AL
L:A$:TAB((28-LEN(A$))/2);RPT$("_",LEN(A$
))
1400 FOR I=1 TO 18 :: IF RUBR$(RD2,I,1)=
"" THEN 1430
1410 J=I+5 :: DISPLAY AT(I+3,1):RUBR$(RD
2,I,1);RPT$(".",19-LEN(RUBR$(RD2,I,1)))
1420 IF MONT(MOIS2,RD2,I)<>0 THEN DISPLA
Y AT(I+3,20):USING 190:MONT(MOIS2,RD2,I)
:: TOTAL(RD2)=TOTAL(RD2)+MONT(MOIS2,RD2,
I)
1430 NEXT I :: DISPLAY AT(J-1,20):RPT$("
_",9):: DISPLAY AT(J,19):USING 200:TOTAL
(RD2)
1440 !entree des montants
1450 FOR I=1 TO 18 :: IF RUBR$(RD2,I,1)=
"" THEN 1490
1460 TOTAL(RD2)=TOTAL(RD2)-MONT(MOIS2,RD
2,I)
1470 ACCEPT AT(I+3,20)VALIDATE(NUMERIC)S
IZE(-9)BEEP:A$ :: IF A$="" THEN MONT(MOI
S2,RD2,I)=0 ELSE MONT(MOIS2,RD2,I)=VAL(A
$)
1480 DISPLAY AT(I+3,20):USING 190:MONT(M
OIS2,RD2,I):: TOTAL(RD2)=TOTAL(RD2)+MONT
(MOIS2,RD2,I):: DISPLAY AT(J,19):USING 2
00:TOTAL(RD2)
1490 NEXT I :: DISPLAY AT(J,1)SIZE(16):"
CORRECT (O/N): O"
1500 ACCEPT AT(J,16)VALIDATE("ON")SIZE(-
1)BEEP:A$ :: IF A$<>"O" THEN CALL HCHAR(
J,3,32,16):: GOTO 1450
1510 !mise a jour fichier
1520 IF MO(RD2,MOIS2)=0 THEN MO(RD2,MOIS
2)=RECN :: RECN=RECN+1
1530 RESTORE #1,REC MO(RD2,MOIS2):: FOR
I=1 TO 17 :: PRINT #1:MONT(MOIS2,RD2,I),
:: NEXT I :: PRINT #1:MONT(MOIS2,RD2,I)
1540 RESTORE #1,REC OFF_AN+RD2-1 :: FOR
I=1 TO 11 :: PRINT #1:MO(RD2,I),:: NEXT
I :: PRINT #1:MO(RD2,I)
1550 RESTORE #1 :: FOR I=1 TO 17 :: PRIN
T #1:AN(I),:: NEXT I :: PRINT #1:RECN ::
 CLOSE #1 :: RETURN
1560 !
1570 ! choix 3: etat des comptes
1580 !
1590 DISPLAY AT(1,6)ERASE ALL:"ETAT DES
COMPTES":TAB(6);RPT$("\",16): :"ANI J F
M A M J J A S O N DI":RPT$("_",28)
1600 OPEN #1:F_MONT$,INPUT ,RELATIVE,INT
ERNAL,FIXED 9*18 :: ANN3=0
1610 FOR I=1 TO 17 :: K=ANN3 :: ANN3=100
 :: OFF_AN=0 :: FOR J=1 TO 17 :: IF AN(J
)>K AND AN(J)<ANN3 THEN ANN3=AN(J):: OFF
_AN=J
1620 NEXT J :: IF OFF_AN=0 THEN DISPLAY
AT(I+5,1):"--|";RPT$(" -",12);"|" :: GOT
O 1670
1630 RESTORE #1,REC OFF_AN*2-1 :: FOR J=
1 TO 2 :: FOR K=1 TO 11 :: INPUT #1:MO(J
,K),:: NEXT K :: INPUT #1:MO(J,K):: NEXT
 J
1640 DISPLAY AT(I+5,1):USING 170:ANN3 ::
 DISPLAY AT(I+5,3):"|"
1650 FOR J=1 TO 12 :: IF MO(1,J)=0 THEN
IF MO(2,J)=0 THEN A$="-" ELSE A$="D" ELS
E IF MO(2,J)=0 THEN A$="R" ELSE A$="~"
1660 DISPLAY AT(I+5,J*2+3):A$ :: NEXT J
:: DISPLAY AT(I+5,28):"|"
1670 NEXT I :: DISPLAY AT(23,1):RPT$("_"
,28):: CLOSE #1 :: ACCEPT AT(24,28)SIZE(
1)BEEP:A$ :: IF A$="P" THEN CALL IMP_ECR
1680 RETURN
1690 !
1700 ! choix 4: bilan
1710 !
1720 DISPLAY AT(1,6)ERASE ALL:"BILAN COM
PTABLE":TAB(6);RPT$("\",15): :"MENSUEL (
1)":"ANNUEL  (2) :";MA4: :"ANNEE (19XX):
"
1730 IF ANN4<>0 THEN DISPLAY AT(7,14):AN
N4
1740 ACCEPT AT(5,15)VALIDATE("12")SIZE(-
1)BEEP:MA4 :: IF MA4=2 THEN 1760
1750 DISPLAY AT(9,1):"MOIS  (1-12):" ::
IF MOIS4<>0 THEN DISPLAY AT(9,14):MOIS4
1760 ACCEPT AT(7,15)VALIDATE(DIGIT)SIZE(
-2)BEEP:A$ :: IF A$="" THEN 1760 ELSE AN
N4=VAL(A$)
1770 DISPLAY AT(24,1):"" :: IF MA4=2 THE
N 1790
1780 ACCEPT AT(9,15)VALIDATE(DIGIT)SIZE(
-2)BEEP:A$ :: IF A$="" THEN 1780 ELSE MO
IS4=VAL(A$):: IF MOIS4<1 OR MOIS4>12 THE
N 1780
1790 OFF_AN=0 :: FOR I=1 TO 17 :: IF AN(
I)=ANN4 THEN OFF_AN=I*2-1
1800 NEXT I :: IF OFF_AN=0 THEN DISPLAY
AT(24,1):"CETTE ANNEE N'EXISTE PAS !!" :
: GOTO 1760
1810 OPEN #1:F_MONT$,INTERNAL,RELATIVE,I
NPUT ,FIXED 9*18
1820 RESTORE #1,REC OFF_AN :: FOR I=1 TO
 2 :: FOR J=1 TO 11 :: INPUT #1:MO(I,J),
:: NEXT J :: INPUT #1:MO(I,J):: NEXT I :
: IF MA4=2 THEN 1890
1830 !calcul bilan mensuel
1840 IF MO(1,MOIS4)=0 AND MO(2,MOIS4)=0
```

```
THEN CLOSE #1 :: DISPLAY AT(24,1):"CE MO
IS N'EXISTE PAS !!" :: GOTO 1760
1850 FOR I=1 TO 2 :: FOR J=1 TO 18 :: MO
NT(MOIS4,I,J)=0 :: NEXT J :: IF MO(I,MOI
S4)=0 THEN 1870
1860 RESTORE #1,REC MO(I,MOIS4):: FOR J=
1 TO 17 :: INPUT #1:MONT(MOIS4,I,J),:: N
EXT J :: INPUT #1:MONT(MOIS4,I,J)
1870 TOTAL(I)=0 :: NEXT I :: A$="BILAN "
&NOM_M$(MOIS4)&" 19"&STR$(ANN4):: K=MOIS
4 :: GOTO 1940
1880 !calcul bilan annuel
1890 FOR I=1 TO 2 :: FOR K=1 TO 18 :: MO
NT(13,I,K)=0 :: NEXT K :: FOR J=1 TO 12
:: FOR K=1 TO 18 :: MONT(J,I,K)=0 :: NEX
T K :: IF MO(I,J)=0 THEN 1920
1900 RESTORE #1,REC MO(I,J):: FOR K=1 TO
 17 :: INPUT #1:MONT(J,I,K),:: MONT(13,I
,K)=MONT(13,I,K)+MONT(J,I,K):: NEXT K
1910 INPUT #1:MONT(J,I,K):: MONT(13,I,K)
=MONT(13,I,K)+MONT(J,I,K)
1920 NEXT J :: TOTAL(I)=0 :: NEXT I :: A
$="BILAN 19"&STR$(ANN4):: K=13
1930 !affichage
1940 CLOSE #1 :: DISPLAY AT(1,(28-LEN(A$
))/2)ERASE ALL:A$:TAB((28-LEN(A$))/2);RP
T$("_",LEN(A$))
1950 FOR I=1 TO 18 :: CALL HCHAR(I+2,2,1
24):: FOR J=1 TO 2 :: IF RUBR$(J,I,1)=""
 THEN 1970
1960 DISPLAY AT(I+2,(J-1)*14+1)SIZE(4):R
UBR$(J,I,2):: DISPLAY AT(I+2,(J-1)*14+5)
:USING 190:MONT(K,J,I):: TOTAL(J)=TOTAL(
J)+MONT(K,J,I)
1970 DISPLAY AT(I+2,J*14):"|" :: NEXT J
:: NEXT I :: CALL HCHAR(21,2,95,29):: CA
LL HCHAR(22,2,124)
1980 DISPLAY AT(22,1):USING 230:TOTAL(1)
,TOTAL(2):: CALL HCHAR(23,2,95,29):: DIS
PLAY AT(24,8):"SOLDE: ";TOTAL(1)-TOTAL(2
)
1990 ACCEPT AT(24,28)SIZE(1)BEEP:A$ :: I
F A$="P" THEN CALL IMP_ECR
2000 RETURN
2010 !
2020 ! choix 5: evolution annuelle
2030 !
2040 RESTORE 320 :: FOR I=128 TO 143 STE
P 4 :: READ A$ :: CALL CHAR(I,A$):: NEXT
 I :: CALL COLOR(13,13,1,14,9,1)
2050 DISPLAY AT(1,5)ERASE ALL:"EVOLUTION
 ANNUELLE":TAB(5);RPT$("\",18): :"ANNEE
(19XX):" :: IF ANN5<>0 THEN DISPLAY AT(4
,14):ANN5
2060 ACCEPT AT(4,15)VALIDATE(DIGIT)SIZE(
-2)BEEP:A$ :: IF A$="" THEN 2060 ELSE AN
N5=VAL(A$)
2070 DISPLAY AT(24,1):"" :: OFF_AN=0 ::
FOR I=1 TO 17 :: IF AN(I)=ANN5 THEN OFF_
AN=I*2-1
2080 NEXT I :: IF OFF_AN=0 THEN DISPLAY
AT(24,1):"CETTE ANNEE N'EXISTE PAS !!" :
: GOTO 2060
2090 OPEN #1:F_MONT$,INTERNAL,RELATIVE,I
NPUT ,FIXED 9*18
2100 RESTORE #1,REC OFF_AN :: FOR I=1 TO
 2 :: FOR J=1 TO 11 :: INPUT #1:MO(I,J),
:: NEXT J :: INPUT #1:MO(I,J):: NEXT I
2110 FOR I=1 TO 2 :: TOTAL(I),MONT(13,I,
19)=0 :: FOR J=1 TO 12 :: MONT(J,I,19)=0
 :: IF MO(I,J)=0 THEN 2140
2120 RESTORE #1,REC MO(I,J):: FOR K=1 TO
 17 :: INPUT #1:MONT(J,I,K),:: MONT(J,I,
19)=MONT(J,I,19)+MONT(J,I,K):: NEXT K ::
 INPUT #1:MONT(J,I,K)
2130 MONT(J,I,19)=MONT(J,I,19)+MONT(J,I,
K):: TOTAL(I)=MAX(TOTAL(I),MONT(J,I,19))
2140 MONT(13,I,19)=MONT(13,I,19)+MONT(J,
I,19):: NEXT J :: NEXT I :: CLOSE #1 ::
A$="EVOLUTION ANNEE 19"&STR$(ANN5)
2150 DISPLAY AT(1,5)ERASE ALL:A$:TAB(5);
RPT$("_",20): :TAB(5);"| RECETTES| DEP
ENSES|":RPT$("_",27)
2160 !evolution en chiffres
2170 FOR I=1 TO 12 :: DISPLAY AT(I+5,1):
USING 210:SEG$(NOM_M$(I),1,3)&".",MONT(I
,1,19),MONT(I,2,19):: NEXT I
2180 DISPLAY AT(18,1):RPT$("_",27):: DIS
PLAY AT(19,1):USING 210:"TOT.",MONT(13,1
,19),MONT(13,2,19)
2190 DISPLAY AT(20,1):RPT$("_",27):: DIS
PLAY AT(21,1):USING 210:"Moy.",MONT(13,1
,19)/12,MONT(13,2,19)/12
2200 DISPLAY AT(22,1):RPT$("_",27):TAB(5
);"SOLDE:" :: DISPLAY AT(23,11):USING 19
0:MONT(13,1,19)-MONT(13,2,19)
2210 ACCEPT AT(24,28)SIZE(1)BEEP:A$ :: I
F A$="P" THEN CALL IMP_ECR
2220 !evolution en batons
2230 CALL HCHAR(4,1,32,21*32):: A$=" [R[
E[C[E[T[T[E[S[ [" :: FOR I=1 TO 10 :: DI
SPLAY AT(I+2,1):SEG$(A$,I*2-1,2);TAB(26)
;"[ [" :: NEXT I
2240 DISPLAY AT(13,1):"0|J_F_M_A_M_J_J_A
_S_O_N_D|_|"
2250 A$=" ]D]E]P]E]N]S]E]S] ]" :: FOR I=
1 TO 10 :: DISPLAY AT(I+13,1):SEG$(A$,I*
2-1,2);TAB(26);"] ]" :: NEXT I
2260 FOR I=1 TO 12 :: J=10*8*MONT(I,1,19
)/MAX(TOTAL(1),TOTAL(2)):: K=INT(J/8)::
IF K<>0 THEN CALL VCHAR(13-K,I*2+3,135,K
)
2270 IF INT(J-8*K)<>0 THEN CALL VCHAR(12
-K,I*2+3,128+INT(J-8*K))
2280 J=10*8*MONT(I,2,19)/MAX(TOTAL(1),TO
TAL(2)):: K=INT(J/8):: IF K<>0 THEN CALL
 VCHAR(14,I*2+3,143,K)
2290 IF INT(J-8*K)<>0 THEN CALL VCHAR(14
```

```
+K,I*2+3,136+INT(J-8*K))
2300 NEXT I :: J=10*8*MONT(13,1,19)/(12*
MAX(TOTAL(1),TOTAL(2)):: K=INT(J/8):: I
F K<>0 THEN CALL VCHAR(13-K,29,135,K)
2310 IF INT(J-8*K)<>0 THEN CALL VCHAR(12
-K,29,128+INT(J-8*K))
2320 J=10*8*MONT(13,2,19)/(12*MAX(TOTAL(
1),TOTAL(2))):: K=INT(J/8):: IF K<>0 THE
N CALL VCHAR(14,29,143,K)
2330 IF INT(J-8*K)<>0 THEN CALL VCHAR(14
+K,29,136+INT(J-8*K))
2340 DISPLAY AT(24,1):USING 240:MAX(TOTA
L(1),TOTAL(2))/10
2350 ACCEPT AT(24,28)SIZE(1)BEEP:A$ :: I
F A$="P" THEN CALL IMP_ECR
2360 RETURN
2370 !
2380 ! choix 6: evolution d'une rubrique
2390 !
2400 RESTORE 380 :: FOR I=128 TO 135 STE
P 4 :: READ A$ :: CALL CHAR(I,A$):: NEXT
 I
2410 DISPLAY AT(1,2)ERASE ALL:"EVOLUTION
 D'UNE RUBRIQUE":TAB(2);RPT$("\",24): :"
RECETTE (1)":"DEPENSE (2)    :";RD6: :"AN
NEE (19XX)   :"
2420 IF ANN6<>0 THEN DISPLAY AT(7,16):AN
N6
2430 ACCEPT AT(5,17)VALIDATE("12")SIZE(-
1)BEEP:RD6 :: IF RD6=1 THEN CALL COLOR(1
3,13,1)ELSE CALL COLOR(13,9,1)
2440 ACCEPT AT(7,17)VALIDATE(DIGIT)SIZE(
-2)BEEP:A$ :: IF A$="" THEN 2440 ELSE AN
N6=VAL(A$)
2450 DISPLAY AT(24,1):"" :: OFF_AN=0 ::
FOR I=1 TO 17 :: IF AN(I)=ANN6 THEN OFF_
AN=I*2-1
2460 NEXT I :: IF OFF_AN=0 THEN DISPLAY
AT(24,1):"CETTE ANNEE N'EXISTE PAS !!" :
: GOTO 2440
2470 DISPLAY AT(15,1):RPT$("_",28):: FOR
 I=1 TO 18 :: IF RUBR$(RD6,I,1)="" THEN
2490
2480 A$=SEG$(RUBR$(RD6,I,1),1,10):: J=IN
T((I-1)/2):: DISPLAY AT(J+16,(I-1-J*2)*1
4+1):USING 170:I,A$
2490 NEXT I :: DISPLAY AT(9,1):"NO DE RU
BRIQUE:" :: IF NR6<>0 THEN DISPLAY AT(9,
16):NR6
2500 ACCEPT AT(9,17)VALIDATE(DIGIT)SIZE(
-2)BEEP:A$
2510 IF A$="" THEN 2500 ELSE NR6=VAL(A$)
:: IF NR6<1 OR NR6>18 THEN 2500 ELSE IF
RUBR$(RD6,NR6,1)="" THEN 2500
2520 OPEN #1:F_MONT$,INTERNAL,RELATIVE,I
NPUT ,FIXED 9*18
2530 RESTORE #1,REC OFF_AN+RD6-1 :: FOR
I=1 TO 11 :: INPUT #1:MO(RD6,I),:: NEXT
I :: INPUT #1:MO(RD6,I)
2540 TOTAL(RD6),MONT(13,RD6,NR6)=0 :: FO
R I=1 TO 12 :: MONT(I,RD6,NR6)=0 :: IF M
O(RD6,I)=0 THEN 2580
2550 RESTORE #1,REC MO(RD6,I):: IF NR6=1
 THEN 2570
2560 FOR J=1 TO NR6-1 :: INPUT #1:MONT(I
,RD6,J),:: NEXT J
2570 INPUT #1:MONT(I,RD6,NR6):: MONT(13,
RD6,NR6)=MONT(13,RD6,NR6)+MONT(I,RD6,NR6
):: TOTAL(RD6)=MAX(TOTAL(RD6),MONT(I,RD6
,NR6))
2580 NEXT I :: CLOSE #1 :: A$=RUBR$(RD6,
NR6,1)
2590 !affichage en chiffres
2600 DISPLAY AT(1,(28-LEN(A$))/2)ERASE A
LL:A$:TAB((28-LEN(A$))/2);RPT$("_",LEN(A
$))
2610 DISPLAY AT(3,9):"ANNEE 19"&STR$(ANN
6):TAB(9);RPT$("_",10): : :TAB(5);RPT$("
_",20)
2620 FOR I=1 TO 12 :: DISPLAY AT(I+7,5):
USING 220:NOM_M$(I),MONT(I,RD6,NR6):: NE
XT I :: DISPLAY AT(20,5):RPT$("_",20)
2630 DISPLAY AT(21,5):USING 220:"TOTAL",
MONT(13,RD6,NR6):: DISPLAY AT(22,5):RPT$
("_",20)
2640 DISPLAY AT(23,5):USING 220:"Moyenne
",MONT(13,RD6,NR6)/12 :: ACCEPT AT(24,28
)SIZE(1)BEEP:A$ :: IF A$="P" THEN CALL I
MP_ECR
2650 !affichage en batons
2660 CALL HCHAR(7,1,32,18*32):: DISPLAY
AT(7,1):RPT$("_",5);RPT$("^",20)
2670 FOR I=1 TO 12 :: DISPLAY AT(I+7,1):
SEG$(NOM_M$(I),1,3);".|" :: IF MONT(I,RD
6,NR6)=0 THEN 2700
2680 J=20*8*MONT(I,RD6,NR6)/TOTAL(RD6)::
 K=INT(J/8):: IF K<>0 THEN CALL HCHAR(I+
7,8,135,K)
2690 IF INT(J-8*K)<>0 THEN CALL HCHAR(I+
7,8+K,128+INT(J-8*K))
2700 NEXT I :: DISPLAY AT(20,1):RPT$("_"
,5);RPT$("^",20):: DISPLAY AT(21,1):"moy
.|" :: IF MONT(13,RD6,NR6)=0 THEN 2730
2710 J=20*8*MONT(13,RD6,NR6)/(12*TOTAL(R
D6)):: K=INT(J/8):: IF K<>0 THEN CALL HC
HAR(21,8,135,K)
2720 IF INT(J-8*K)<>0 THEN CALL HCHAR(21
,8+K,128+INT(J-8*K))
2730 DISPLAY AT(22,1):RPT$("_",5);RPT$("
^",20):: DISPLAY AT(24,1):USING 240:TOTA
L(RD6)/20
2740 ACCEPT AT(24,28)SIZE(1)BEEP:A$ :: I
F A$="P" THEN CALL IMP_ECR
2750 RETURN
2760 !
2770 !impression ecran
2780 !
2790 SUB IMP_ECR
2800 DISPLAY AT(24,1):"FONCTION NON REAL
ISEE" :: ACCEPT AT(24,28)SIZE(1)BEEP:A$
2810 SUBEND
```

Mission spéciale

Edouard Krok

N.D.L.R. : ce jeu d'aventure est écrit en ***Basic Etendu****.*

Des diamants ont été dérobés par l'infâme prince Krapul, qui les a cachés dans son château perdu au fin fond du désert ! Personne ne sait où il se trouve exactement et les rares audacieux qui ont osé partir à sa recherche n'en sont jamais revenus. Serez-vous volontaire pour cette mission bien spéciale qui consiste à retrouver et à ramener les diamants volés ?

Si vous avez un goût prononcé pour l'aventure, la découverte et l'imprévu, stoppez là votre lecture et partez immédiatement à la recherche du château.

Si vous préférez vous faire une idée des petites misères qui vous attendent, quelques explications vous seront peut-être utiles.

Au début, vous suivez un petit chemin de campagne et vous décidez de la direction à prendre en appuyant sur l'une des quatre touches fléchées. Votre premier objectif sera de trouver le passage qui mène au désert et à ses pistes peu visibles où votre sens de l'orientation sera mis à rude épreuve. Dès que vous aurez repéré le château, n'hésitez pas à y pénétrer : il est peu probable que vous ayez besoin d'explorer les 25 salles qui le composent pour trouver les diamants. A ce moment, votre but sera de repartir en ville.

Pour cela, vous pourrez tout naturellement refaire en sens inverse le chemin qui vous a conduit au château en retraversant le désert, puis la campagne pour revenir à votre point exact de départ où il vous sera possible d'entrer dans la ville, si vous avez au moins un diamant sur vous. Ce retour vous sera grandement facilité si vous avez eu la sagesse de faire un plan des zones que vous avez visitées. Une autre solution, plutôt inattendue, consiste à utiliser les souterrains du château qui, si vous ne vous y égarez pas, vous conduiront directement en ville où il vous faudra encore retrouver votre rue. En espérant que vous n'aurez pas perdu quelques diamants en cours de route, car il ne vous resterait plus qu'à repartir !

Mais n'allez tout de même pas vous imaginer que le prince Krapul va se laisser tranquillement dépouiller : ses acolytes, de monstrueuses créatures qui crachent des flammes mortelles, vont essayer de vous décourager et, au besoin, de vous détruire. Pour avoir une chance de réussir, vous devez donc être armé; c'est pour cette raison que vous recevrez, avant de partir, une grenade et un pistolet. D'ailleurs, si au cours de votre mission vous perdiez tous vos objets, vous pourriez essayer de revenir à votre point de départ où vous percevriez ces deux armes. Rien ne vous empêche d'ailleurs de laisser les objets que vous transportez dans la campagne avoisinante et de les reprendre au passage.

Si vous rencontrez un ennemi, vous pouvez l'éliminer radicalement avec votre grenade ou tenter de vous défendre avec votre pistolet que vous déplacez de droite à gauche avec les deux touches fléchées "D" et "S".

Le tir se fait en appuyant sur la touche "1" mais, comme il ne s'agit pas d'un pistolet mitrailleur, vous serez obligé de relâcher la gachette (la touche "1") avant de tirer à nouveau. Ces créatures ont la peau dure et il faut souvent les toucher à plusieurs reprises avant qu'elles ne cèdent. Le fantôme, quant à lui, lorsqu'il est atteint, se transforme en une créature différente qui vous attaque aussitôt. Si vous remportez le combat, l'ennemi disparaît définitivement de cet endroit. Si vous perdez, il reste en place, votre pistolet est détruit mais vous pouvez tout de même poursuivre votre route. Si vous n'avez plus de pistolet et que vous rencontriez un ennemi, vous pourrez passer à condition de lui offrir un objet qu'il se choisira lui-même parmi ceux que vous transportez. Ce serait un grand malheur si vous n'aviez plus rien à lui présenter car, dans ce cas, vous n'auriez plus aucun espoir de rentrer un jour chez vous !

Fort heureusement, vous trouverez au hasard de votre voyage des objets qui s'avèreront très utiles. Vous pourrez les prendre, les laisser, voire les déplacer. Si vous avez une clochette, vous l'agiterez chaque fois que vous rencontrerez un serpent : il ira vite se terrer dans son trou et n'en ressortira que lorsque vous vous serez éloigné. Il vous arrivera de trouver un coffre; vous ne pourrez le prendre car il est trop lourd à porter mais ce sera un jeu d'enfant de l'ouvrir si vous avez la clé. La plupart des coffres renferment un diamant mais, s'il peuvent être vides où contenir un objet utile, un ennemi peut aussi s'y cacher.

Si vous avez de la chance, vous trouverez une carte; son nom est usurpé car il n'existe en fait aucun plan de ces zones inexplorées, mais une lettre repérant le lieu où vous vous trouvez s'inscrira en haut de l'écran et, si vous le demandez, vous obtiendrez la lettre qui repère l'endroit par où vous êtes entré et celui par lequel vous devez sortir. Contrairement aux premières apparences, la carte est très utile pour se guider dès que l'on s'en sert régulièrement. Les jumelles, quant à elles, vous feront gagner beaucoup de temps car elles permettent de regarder dans toutes les directions sans prendre le risque d'aller voir sur place.

Si vous êtes totalement égaré, que vous vous retrouviez seul, sans arme et sans moral, il vous reste encore la faculté d'abandonner. Si vous tapez "REDO" (FCTN-8) vous pourrez repartir en mission avec un parcours exactement identique, les objets et les ennemis vous attendants aux mêmes endroits. Ainsi, si vous avez fait un plan sur papier, il pourra vous resservir. Si vous tapez "BACK" (FCTN-9) vous partirez chaque fois dans l'inconnu le plus total.

Avant le départ, vous recevez votre ordre de mission puis vous devez vous engager à ramener entre 1 et 5 diamants; cela conditionne la difficulté de votre tentative. Avec un peu d'expérience, vous constaterez que les problèmes qui se posent pour apporter les 5 diamants à la fois, ne sont pas insurmontables. □

```
100 ! ********************
110 ! * MISSION SPECIALE *
120 ! ********************
130 ! *   basic etendu   *
140 ! ********************
150 !
160 DISPLAY AT(7,7)ERASE ALL:"mi
ssion speciale" :: CALL MAGNIFY(
4):: D=1 :: B=2
:: F=8
170 DIM P$(4),A(3),N(3)
180 DEF T(X)=ASC(SEG$(P$(P),X,D)
)
190 CALL CHAR(33,"00FF7060600000
00382472FAF8F8F870FE82AA92AA82FE
FE")
200 CALL CHAR(36,"0000C0C6FFC000
0010101038387CFE106666FFE7E7E77E
E738FE7C38381")
210 FOR X=64 TO 88 STEP F :: CAL
L CHAR(X,"FFFEFCF8F0E0C080FF7F3F
1F0F070301"&RPT$
("F",16)&"0")
220 CALL CHAR(X+4,"FFFFFFFFFF3F0
F03FF3F0F0300000000FFFFFFFFFFCF
0C0FFFCF0C"):: N
EXT X
230 CALL CHAR(90,"000000000030F
3F0000000000C0F0FC",68,"02020202
060600000",70,"4
040404060600000")
240 CALL CHAR(56,"808080000008080
8001010100000010101",132,"0101"&R
PT$("0",45))
250 CALL CHAR(128,"F0F0E0E0C0C08
080FFFFFEFEFCFCF8F80F0F070703030
101FFFF7F7F3F3F1
F1F")
260 CALL CHAR(124,"0102020604050
8080911111108081122008040406020A
0909018088888844
422")
270 CALL DELSPRITE(ALL):: CALL C
OLOR(D,5,D,4,B,D):: IF K=15 THEN
 RANDOMIZE ELSE
RANDOMIZE 12
280 DISPLAY AT(B,D)ERASE ALL:"vo
tre mission :"
290 DISPLAY AT(7,5):"vous devez
recuperer et": :"ramener chez vo
us en ville": :"
les diamants derobes par": :"l`i
nfame prince krapul ;"
300 X$="IMLILKONJJCGNKFIMONHCGGG
FIEMELCMNIFIFKGLJIOMNCFCGFIELILC
LJKNINCOFJKMGLCG
GEFILHILCGGONIMLKFKNKOLCFCGFHIEL
HJCMFJKEOMFKMGOLCGEGF"
310 Q$="@@@AAABBCDEFHHHIIJJKKLL"
320 FOR P=C TO 4 :: X=INT(25*RND
)+D :: Y=INT(25*RND)+D :: IF X=Y
 THEN 320 ELSE P
$(P)=SEG$(X$,INT(4*RND)*25+D,25)
&CHR$(X)&CHR$(Y)
330 FOR X=D TO 25 :: P$(P)=P$(P)
&SEG$(Q$,INT(23*RND)+D,D):: NEXT
 X :: NEXT P
340 DISPLAY AT(17,5)BEEP:"combie
n de diamants": :"vous sentez vo
us capable": :"d
e ramener :": : :TAB(8);"choisir
  1 a 5"
350 CALL KEY(3,K,S):: IF K<49 OR
 K>53 THEN 350 ELSE CALL SOUND(9
9,999,D):: DISPL
AY AT(22,7)ERASE ALL:"preparez v
ous": :"vous allez a la campagne
"
360 Q$="@@@@" :: Q=K-48 :: FOR X
=C TO Q :: CALL MAJ(P$(B),INT(25
*RND)+28,71):: N
EXT X
370 FOR P=C TO 4 :: FOR Y=D TO B
 :: CALL MAJ(P$(P),T(25+Y)+27,64
):: NEXT Y :: NE
XT P :: DISPLAY AT(D,16):"1 2 3
4 5" :: CALL HCHAR(D,F,39,Q):: P
,W=C
380 CALL DELSPRITE(ALL):: GOSUB
1530 :: P=P+W :: E=T(26-(W<C))+6
4 :: Q$=Q$&"@" :
: IF P=3 THEN 390 ELSE IF P=B TH
EN GOSUB 1290 ELSE GOSUB 1010
390 FOR X=C TO 3 :: A(X)=D :: N(
X)=C :: NEXT X :: GOTO 410
400 GOSUB 1530 :: W=T(E)-64 :: F
OR X=3 TO C STEP -D :: A(X)=N(X)
:: N(X)=-(W>=B^X
):: W=W-N(X)*B^X :: NEXT X
410 FOR K=-D TO B :: S=N(K+D)::
IF K<B THEN IF S=A(K+D)THEN 440
ELSE 430 ELSE IF
 P=3 THEN GOSUB 1460 ELSE IF P=B
 THEN GOSUB 1340 ELSE GOSUB 1250
420 GOTO 440
430 IF P=B AND K=C THEN GOSUB 13
20 ELSE IF P=B THEN GOSUB 1360 E
LSE IF P=3 THEN
GOSUB 1400 ELSE GOSUB 1070
440 NEXT K :: IF E>64 THEN E=E-6
4 :: GOTO 400 ELSE Z=T(E+27)-64
450 IF Z<F THEN CALL CHARPAT(32+
Z,X$):: CALL CHAR(60,X$&RPT$("0"
,64))ELSE GOSUB
1490
460 CALL SCREEN(ASC(SEG$("NJPHD"
,P+D,D))-64):: IF POS(Q$,"C",D)T
HEN CALL HCHAR(D
```

```
,14,96+E)
470 ON P+D GOSUB 1540,1560,1570,
1580,1590
480 CALL SPRITE(#4,60,5-3*(P=B)*
(Z>F),129+24*(Z>F),116):: IF LEN
(Q$)>5 THEN Q$=S
EG$(Q$,D,5):: IF Z>F THEN 660 EL
SE 540 ELSE IF Z>F THEN 660
490 W=(E=T(26))-(E=T(27)):: ON W
+B GOTO 510,540,500
500 IF P=4 THEN CALL SCREEN(13):
: CALL TEX(5):: IF Q>C THEN DISP
LAY AT(24,B):"al
lez chercher les diamants" :: GO
TO 550 ELSE 960 ELSE 520
510 IF P=C THEN IF Q$="@@@@@" TH
EN CALL MAJ(Q$,D,65):: CALL MAJ(
Q$,B,66):: GOTO
540 ELSE IF POS(Q$,"G",D)=C THEN
 540 ELSE W=4
520 CALL TEX(P+W)
530 CALL KEY(C,K,S):: IF K=79 TH
EN CALL HCHAR(22,D,32,15):: DISP
LAY AT(22,10):"v
ous allez" :: GOTO 380 ELSE IF K
<>78 THEN 530
540 IF Z>C AND Z<9 THEN X$=" un
objet" :: DISPLAY AT(21,D)BEEP:"
 ":"      p  prend
re";X$:"  1a5  utiliser";X$:"l 1
a5  laisser";X$ ELSE CALL TEX(7)
550 CALL KEY(C,K,S):: IF S THEN
CALL SOUND(99,999,C)ELSE 550
560 IF K>48 AND K<54 THEN 840
570 IF K=80 THEN X=POS(Q$,"@",D)
:: IF X=C OR Z>7 THEN 650 ELSE C
ALL DELSPRITE(#4
):: CALL MAJ(Q$,X,Z+64):: CALL M
AJ(P$(P),E+27,64):: Q=Q+(Z=7)::
Z=C :: GOTO 540
ELSE IF K<>76 OR Z>F THEN 600
580 CALL KEY(C,K,S):: IF K>48 AN
D K<54 THEN Z=ASC(SEG$(Q$,K-48,D
))-64 ELSE 580
590 IF Z THEN CALL MAJ(Q$,K-48,6
4):: CALL MAJ(P$(P),E+27,Z+64)::
 Q=Q-(Z=7):: GOT
O 450 ELSE 650
600 IF K=68*N(C)THEN E=E+D :: GO
TO 400
610 IF K=69*N(D)THEN E=E-5 :: GO
TO 400
620 IF K=83*N(B)THEN E=E-D :: GO
TO 400
630 IF K=88*N(3)THEN E=E+5 :: GO
TO 400
640 IF K=15 OR K=6 THEN 270
650 FOR X=15 TO C STEP -D :: CAL
L SOUND(400-25*X,150+90*X,X):: N
```

```
EXT X :: Z=INT(4
*RND+9):: GOTO 450
660 FOR X=C TO 250 STEP 50 :: CA
LL SOUND(400,400-X,6,800+3*X,3):
: CALL SOUND(500
,111,30):: NEXT X :: X=POS(Q$,"A
",D):: IF X THEN 690 ELSE IF Q$=
"@@@@@" THEN 970
670 Y=INT(5*RND)+D :: X$=SEG$(Q$
,Y,D):: IF X$="G" THEN Q=Q+D
680 IF X$="@" THEN 670 ELSE CALL
 MAJ(Q$,Y,64):: GOTO 490
690 CALL MAJ(Q$,X,64):: CALL SPR
ITE(#3,68,B,149,90+76*INT(B*RND)
,#D,124,9,D,256,
#B,132,B,D,256,-25,C):: FOR X=C
TO 1000*RND
700 CALL KEY(C,K,S):: IF S THEN
IF K>48 AND K<54 THEN Y=ASC(SEG$
(Q$,K-48,D))-64
:: IF Y=B OR Y=5 THEN 850 ELSE 7
20 ELSE 720
710 NEXT X
720 ON ERROR 770
730 CALL KEY(C,K,S):: W=K*S=49 :
: CALL POSITION(#4,X,X,#3,Y,Y)::
 S=(ABS(X-Y)<12)
*(W=C)
740 IF W-S=C THEN CALL MOTION(#3
,C,26*(K=68)*(Y<165)-26*(K=83)*(
Y>70),#4,C,(Z-F)
*9*SGN(Y-X)):: CALL LOCATE(#D,D,
256,#B,D,256):: GOTO 730
750 CALL POSITION(#4,X,X,#3,Y,Y)
:: CALL LOCATE(#B,145,256+(256-Y
)*W,#D,125,256-(
256-X)*S,#9+5*W,105,4*Z*RND+X-Z-
Z)
760 CALL SOUND(-90,-S-5,C):: CAL
L COINC(#4,#B,23,W):: CALL COINC
(#3,#D,24,S):: O
N D-W-W-S GOTO 730,780,800
770 RETURN 720
780 CALL SOUND(300,-D,C):: CALL
MOTION(#3,C,C,#4,C,C):: CALL DEL
SPRITE(#B,#D)::
CALL COLOR(#3,16)
790 FOR X=C TO 220 :: CALL SCREE
N(10-6*(X>20)):: NEXT X :: GOTO
830
800 CALL SOUND(300,-D,C):: CALL
MOTION(#4,C,C,#3,C,C):: IF Z=10
THEN Z=INT(4*RND
)+9 :: GOSUB 1490 :: GOTO 720
810 FOR X=D TO 12 :: CALL COLOR(
#4,9,#B,D):: CALL SOUND(10,-F,X/
B):: CALL COLOR(
#4,1):: CALL SOUND(10,-F,X/B)::
NEXT X :: Z=C
```

```
820 CALL POSITION(#3,X,Y):: W=PO
S(Q$,"@",D):: IF X*W THEN CALL M
AJ(Q$,W,65)
830 CALL DELSPRITE(#3,#B):: CALL
 MAJ(P$(P),E+27,64+Z):: GOTO 490
840 Y=ASC(SEG$(Q$,K-48,D))-64
850 ON Y+D GOTO 650,650,860,870,
880,890,910,950
860 CALL SOUND(300,-7,C):: CALL
MAJ(Q$,K-48,64):: GOTO 810
870 CALL TEX(6):: FOR X=C TO D :
: CALL HCHAR(22+B*X,21,T(26+X)+9
6):: NEXT X :: G
OTO 550
880 IF Z<>F THEN 650 ELSE X=INT(
26*RND):: Z=X+(X-7)*(X>12):: CAL
L SOUND(200,500,
D):: CALL SOUND(150,900,D):: CAL
L MAJ(P$(P),27+E,64+Z):: GOTO 45
0
890 FOR X=-D TO 3*RND :: CALL SO
UND(400,784,C):: CALL SOUND(400,
585,3):: CALL SO
UND(800,784,3):: NEXT X
900 IF Z=9 THEN CALL DELSPRITE(#
4):: GOTO 820 ELSE IF Z>9 THEN 7
20 ELSE 550
910 CALL TEX(9):: FOR X=C TO 3 :
: X$="rien" :: IF N(X)THEN W=E-(
X=C)+(X=B)+5*(X=
D)-5*(X=3)ELSE 940
920 IF W=T(26)OR W=T(27)THEN X$=
"sortie" :: GOTO 940 ELSE Y=T(27
+W)-64
930 IF Y THEN IF Y<F THEN X$=CHR
$(32+Y)ELSE IF Y=F THEN X$="coff
re" ELSE X$="enn
emi"
940 DISPLAY AT(21+X-B*(X=C),20-5
*X-10*(X=3)-LEN(X$)/B)BEEP SIZE(
6):X$ :: NEXT X
:: GOTO 550
950 CALL MAJ(Q$,K-48,64):: Q=Q+D
 :: GOTO 550
960 CALL SCREEN(B):: DISPLAY AT(
23,7):"mission accomplie" :: FOR
 X=D TO 4 :: FOR
 Y=150*X TO 500*X STEP 50 :: CAL
L SOUND(60,Y,C):: NEXT Y :: NEXT
 X :: CALL SCREE
N(13):: GOTO 980
970 CALL TEX(F):: FOR X=22 TO D
STEP -D :: CALL SOUND(500-X*X,11
0+12*X,X/B):: NE
XT X
980 FOR X=C TO 990 :: NEXT X
990 CALL SOUND(40,2000,9):: DISP
LAY AT(24,4):"tapez  `redo` ou `
back`" :: FOR X=
C TO 30 :: CALL KEY(C,K,S):: IF
K=15 OR K=6 THEN CALL SOUND(200,
800,B):: GOTO 27
0
1000 NEXT X :: DISPLAY AT(24,D):
"" :: GOTO 990
1010 FOR X=C TO F :: CALL HCHAR(
3+X,3,82,28):: CALL HCHAR(12+X,3
,66,28):: NEXT X
 :: CALL HCHAR(12,3,74,28)
1020 IF P<4 THEN 1040 ELSE CALL
CHAR(40,"0001031333330000333130
3010000008080808
0808000008080808080808")
1030 CALL CHAR(44,"10181C1C1C1C0
0001C1C1C1C1810000000000000080C0C00
000C0C08"):: RET
URN
1040 IF P THEN CALL CHAR(40,"010
A25174F3F5FBF5F3F4F17250A010000A
048D0E4F8F4FAF4F
8E4D048A"):: CALL SPRITE(#5,40,D
,25,80):: RETURN
1050 CALL CHAR(40,"21777FFFFF3F3
E0C0",42,"88D8FEFFFFFEFE640",44,
"81C67F070E18000
00",46,"8040E0C",47,"")
1060 CALL SPRITE(#5,40,D,36,D,C,
-4,#6,40,D,26,D,C,-B,#7,40,D,23,
10,C,-B,#F,44,D,
30,D,C,B):: RETURN
1070 Z=12-(K=-D)-9*K :: Y=ABS(K)
:: IF S-Y THEN 1130 ELSE CALL HC
HAR(20,13-10*K,6
7,F*S):: CALL HCHAR(20,16.5-5.5*
K,129-K,S)
1080 FOR X=16 TO 19 :: CALL HCHA
R(X,11-F*K,31,12*S):: CALL HCHAR
(X-4,Z,74,10-S):
: NEXT X
1090 CALL HCHAR(15,Z,67,10-S)::
FOR X=13-1.5*P TO -14*(P<>D):: C
ALL HCHAR(X,Z,66
,10-S):: NEXT X
1100 IF P<4 THEN CALL HCHAR(11,Z
,82,-F*(K=C)):: RETURN ELSE IF K
=C THEN CALL DEL
SPRITE(#5,#7)
1110 FOR X=3 TO 6 :: CALL HCHAR(
X,Z,82,10-S):: NEXT X
1120 FOR Y=B TO 6 STEP 4 :: FOR
X=C TO D :: CALL VCHAR(9,Z+Y+X+(
K=D),74,6):: NEX
T X :: NEXT Y :: CALL HCHAR(11,Z
,66,9):: RETURN
1130 FOR X=13+10*(P=4)TO 20*Y ::
 CALL HCHAR(X,Z,66,9+S):: NEXT X
 :: FOR X=12 TO
15*S :: CALL HCHAR(X,Z,66,10)::
```

```
NEXT X
1140 FOR Z=K+(K=C)TO K-(K=C)STEP
 B :: W=Z=D :: FOR X=3*Y TO B+6*
Y+6*(P=D):: CALL
 HCHAR(12+X,15-X*Z,67,4):: CALL
HCHAR(12+X,17+W-B*Z-Z*X,65+W)::
NEXT X
1150 FOR X=Y*(B-N(D)/B)TO 1.5+2.
5*Y STEP .5 :: CALL HCHAR(12+B*X
,18+3.5*W-X*Z,12
9+X-INT(X)-Z):: NEXT X
1160 IF P=4 THEN 1190 ELSE IF P
THEN CALL HCHAR(16,21+9*W,66,D-S
):: GOTO 1240
1170 FOR X=D*Y TO .5+4.5*Y STEP
.5 :: CALL HCHAR(11.5+X,20+W*7-B
*X*Z,77.5+Z-X+IN
T(X)):: CALL HCHAR(11.5+X,3-(19+
B*INT(X))*(Z=-D),74,9-B*INT(X)-(
X=5)):: NEXT X
1180 CALL HCHAR(11,16.5-2.5*Z,90
.5-.5*Z,S):: GOTO 1240
1190 FOR X=3*Y TO B+4*Y :: CALL
HCHAR(9-X,19+5*W-X*Z,80-W):: NEX
T X :: FOR X=D T
O 4+3*S :: CALL HCHAR(B+X,F+X,82
,18-B*X):: NEXT X
1200 IF S THEN CALL HCHAR(10,14,
66,6):: CALL HCHAR(10,15,82,4)::
 CALL HCHAR(11,1
5,82,4):: CALL SPRITE(#6+Z,42+Z+
Z,D,68,116-Z*27):: GOTO 1240
1210 FOR X=C TO B :: CALL VCHAR(
3,29+25*W+X*Z,74,18):: NEXT X ::
 CALL HCHAR(3,16
.5-10.5*Z,88-W):: CALL HCHAR(20,
16.5-10.5*Z,73+W)
1220 FOR X=D TO B :: C=17+W-8*Z+
X*Z :: CALL VCHAR(7,C,74,10):: C
ALL HCHAR(5+X,C,
88-W):: CALL HCHAR(18-X,C,73+W):
: NEXT X :: C=0
1230 CALL HCHAR(11,23+19*W,66,7)
1240 NEXT Z :: RETURN
1250 FOR X=4-4*N(D)TO F :: CALL
HCHAR(12+X,15.5-X/B,31,B+X+X/B-I
NT(X/B)):: NEXT
X :: CALL HCHAR(12,18,130,N(D)):
: CALL HCHAR(20,22,130,S)
1260 CALL HCHAR(20,11,128,S):: C
ALL HCHAR(20,11,67,12-12*S):: CA
LL HCHAR(15,12,6
7):: CALL HCHAR(15,21,67):: IF P
=4 THEN 1280 ELSE IF P THEN RETU
RN
1270 FOR X=-D TO D STEP B :: CAL
L VCHAR(13,16.5+(4.5+N(D))*X,56.
5+X/2-N(D)*X,-B*
(N(D)=N(D-X))):: NEXT X :: RETUR
N
1280 CALL VCHAR(7,22,56,-F*(N(C)
=N(D))):: CALL VCHAR(7,11,57,-F*
(N(D)=N(B))):: R
ETURN
1290 FOR X=3 TO 20 :: CALL HCHAR
(X,3,74,28):: NEXT X :: FOR X=D
TO 5 :: X$=RPT$(
"B",4+4*X):: DISPLAY AT(F-X,11-X
*B)SIZE(F+X*4):"]\"&X$&"^_"
1300 DISPLAY AT(15+X,11-X*B)SIZE
(F+X*4):"NO"&X$&"ML" :: NEXT X :
: CALL HCHAR(17,
14,67,6):: CALL HCHAR(18,13,67,F
)
1310 FOR X=C TO 3 :: CALL CHAR(4
0+X,RPT$("0",13-4*X)&RPT$("F",3+
4*X)):: CALL CHA
R(44+X,RPT$("0",4*X)&"F0"&RPT$("
F",14-4*X)):: NEXT X :: RETURN
1320 FOR X=F TO -15*(S=C):: CALL
 HCHAR(X,13,31,F):: NEXT X :: DI
SPLAY AT(16,12)S
IZE(6):"@CCCCA" :: CALL HCHAR(16
,14,66,-6*(S=C))
1330 FOR X=15 TO 18*S :: CALL VC
HAR(10,X,82,6):: NEXT X :: RETUR
N
1340 FOR X=19 TO -20*(S=C):: CAL
L HCHAR(X,10,66,14):: NEXT X
1350 FOR X=D TO B*S :: X$=RPT$("
C",F+B*X):: DISPLAY AT(18+X,10-X
)SIZE(10+B*X):"@
"&X$&"A" :: NEXT X :: RETURN
1360 W=K=D :: FOR X=C TO D :: FO
R Y=C TO D :: Z=17+W-6*K-(B*X+Y)
*K :: CALL VCHAR
(9,Z,74+S*F,9):: CALL HCHAR(17+X
,Z,78+K+F*S-Y):: NEXT Y
1370 FOR Y=D TO 3+X :: CALL HCHA
R(17+X,17+W-(B+X+Y)*K,66+S):: NE
XT Y :: CALL HCH
AR(17+X,17+W-(3+X)*K,65+W,-(S=C)
):: NEXT X
1380 IF S THEN IF W THEN DISPLAY
 AT(9,5)SIZE(4):",-./" ELSE DISP
LAY AT(9,21)SIZE
(4):"()*+"
1390 RETURN
1400 Z=12-(K=-D)-9*K :: Y=S+ABS(
K)=D :: FOR X=C TO F-(K=0):: CAL
L VCHAR(3,Z+X,74
-Y,14+4*ABS(K)):: NEXT X :: IF Y
 THEN 1420
1410 FOR X=6 TO F :: CALL HCHAR(
X-F*(X=F),Z,66,9-(K=C)):: CALL H
CHAR(X+11,Z+(K=-
D),67,10*ABS(K)):: NEXT X :: CAL
L HCHAR(20,30+(K=D)*27,75,S):: R
```

```
ETURN
1420 W=N(D)=C :: FOR X=-3*W TO 5
 :: CALL HCHAR(F-X,15-X,74,4+B*X
):: NEXT X :: Y=
ABS(K):: FOR Z=K+(K=C)TO K-(K=C)
STEP B
1430 FOR X=3*Y TO B+3*Y :: W=19+
(Z=D)*5-Z*X :: CALL HCHAR(F-X,W,
72-(Z=D)):: CALL
 HCHAR(15+X,W,65+(Z=D)):: NEXT X
1440 FOR X=1.5*Y TO 1+4.5*Y STEP
 .5 :: W=19+(Z=D)*5-B*X*Z :: CAL
L HCHAR(8.5-X,W,
87+(Z=D)*B+(X>INT(X))):: CALL VC
HAR(9.5-X,W,74,5+B*X)
1450 CALL HCHAR(14+X,W,76-(Z=D)*
B-(X>INT(X))):: NEXT X :: NEXT Z
 :: FOR X=15 TO
18*S :: CALL VCHAR(9,X,31,6):: N
EXT X :: RETURN
1460 CALL HCHAR(20,4,66,26):: FO
R X=B-B*N(D)TO 4+S :: CALL HCHAR
(15+X,15-X,67,4+
B*X):: NEXT X
1470 CALL HCHAR(20,9,64,S):: CAL
L HCHAR(20,24,65,S):: CALL HCHAR
(17,21,65+B*N(C)
):: CALL HCHAR(17,12,64+3*N(B))
1480 CALL VCHAR(F,11,57,F+F*(N(D
)+N(B)=D)):: CALL VCHAR(F,22,56,
F+F*(N(C)+N(D)=D
)):: RETURN
1490 DATA 3F7F7700777F7F7F7F0000
0000000000FCFEEE00EEFEFEFEFE
1500 DATA 020207070A0F0702020203
01000F3FFF0000000080800000000000
80C0F0FCFF,03078
5C7673F3F1F0F0F07070303010180C04
0C1C7FEFCF8F8F0F0E0E0C0C08
1510 DATA 03050702011F3F7FDF8F87
C3860C08388040C08000F0F8FCF6E2C2
87C2602038,030E1
02405030507020F1B31F0000000E0380
424A2C2A2E448F8D88C0F
1520 RESTORE :: FOR X=F TO Z ::
READ X$ :: NEXT X :: CALL CHAR(6
0,X$):: RETURN
1530 CALL HCHAR(D,14,32):: Y=INT
(15*RND)+B :: CALL COLOR(C,Y,Y,B
,Y,Y,13,Y,Y):: F
OR X=4 TO F :: CALL COLOR(#X,D):
: CALL COLOR(X,Y,Y):: NEXT X ::
RETURN
1540 CALL COLOR(C,15,15,4,B,13,5
,13,12,6,4,13,7,F,D,F,13,F,13,12
,15)
1550 CALL COLOR(#5,16,#6,15,#7,1
5,#F,9):: RETURN
1560 CALL COLOR(C,11,11,5,12,12,
6,12,D,7,F,D,13,12,11):: CALL CO
LOR(#5,12):: RET
URN
1570 CALL COLOR(C,6,6,B,F,5,5,14
,12,6,5,14,7,F,12,F,14,5):: RETU
RN
1580 CALL COLOR(C,B,B,4,12,7,5,1
1,12,6,7,11,7,11,7):: RETURN
1590 CALL COLOR(C,15,15,4,B,10,5
,10,14,6,12,14,7,F,10,F,10,12,13
,14,15):: CALL C
OLOR(#5,12,#7,12):: RETURN
1600 SUB TEX(A)
1610 CALL HCHAR(21,1,32,128):: I
F A=9 THEN SUBEXIT ELSE IF A<5 T
HEN DISPLAY AT(2
2,1)BEEP:"o n  desirez vous alle
r"
1620 ON A+1 GOTO 1630,1640,1650,
1660,1670,1680,1690,1700,1710
1630 DISPLAY AT(24,9):"a la camp
agne" :: SUBEXIT
1640 DISPLAY AT(24,6):"affronter
 le desert" :: SUBEXIT
1650 DISPLAY AT(24,6):"visiter l
e chateau" :: SUBEXIT
1660 DISPLAY AT(24,5):"dans les
souterrains" :: SUBEXIT
1670 DISPLAY AT(24,11):"en ville
" :: SUBEXIT
1680 DISPLAY AT(22,2)BEEP:"vous
etes arrive chez vous" :: SUBEXI
T
1690 DISPLAY AT(22,11):"entree":
 :TAB(11);"sortie" :: SUBEXIT
1700 DISPLAY AT(21,1)BEEP:"depla
cement":TAB(9);"par touches flec
hees":"tir par t
ouche 1":"abandon  `redo` ou `ba
ck`" :: SUBEXIT
1710 DISPLAY AT(22,2):"Krapul a
eu raison de vous":"   ici sera
 votre tombe"
1720 SUBEND
1730 SUB MAJ(A$,R,C)
1740 A$=SEG$(A$,1,R-1)&CHR$(C)&S
EG$(A$,R+1,99):: IF LEN(A$)>6 TH
EN SUBEXIT
1750 IF C=64 THEN CALL SOUND(500
,250,5)ELSE CALL SOUND(300,900,4
)
1760 CALL HCHAR(1,17+2*R,C-32)
1770 SUBEND
```

Crayon optique

Bernard Bezuel

Ce programme, écrit en assembleur, permet de créer des dessins directement à l'écran à l'aide d'un crayon optique. Cet accessoire (qu'il vous faudra construire en suivant les indications jointes à cet article) n'est pas indispensable car le programme peut aussi recevoir des ordres provenant du clavier.

Les fichiers source qui apparaissent pages suivantes concernent une version utilisable avec une extension de mémoire 32Ko. Cependant, nos lecteurs munis seulement de la Mini-mémoire pourront utiliser ce programme en entrant, sous EASY BUG, la liste de codes machines placée en fin d'article (une autre solution, plus simple, consiste à charger le code objet situé sur la cassette d'accompagnement du numéro 8, avec l'option L de EASY BUG; le programme est baptisé "CRAYON"). La différence entre les deux versions porte essentiellement sur l'affichage des menus, qui ont dus être raccoucis afin de permettre leur implantation dans les 4Ko de la Mini-mémoire. Pour la version "éditeur/assembleur", nous avons prévu d'autres fonctions sous forme de modules qui seront ajoutés par la suite; cela ne sera bien sûr pas possible avec la version "Mini-mémoire".

Nous ne nous étendrons pas sur le fonctionnement du programme car les nombreuses remarques présentes dans les listings vous permettront de comprendre aisément. De même, il n'est pas nécessaire de donner un mode d'emploi pour la version "éditeur/assembleur", les divers menus étant très "parlants". Ce n'est pas tout à fait aussi simple avec la version "Mini-mémoire"; nous vous proposons donc un petit mode d'emploi dans lequel les chiffres représentent les touches qu'il faut presser pour accéder à une fonction; les lettres minuscules entre parenthèses correspondent aux mêmes fonctions, mais cette fois avec le crayon optique.

Menu principal

1 – (p) : point
2 – (t) : trait
3 – (d) : droite
4 – (c) : curviligne
5 – (e) : écran

Options

1 : Point

Mise en place d'un point

2 : Trait

1 (v) visible
le déplacement de la mire laisse une trace visible (pixels allumés).

2 (i) invisible
efface les points visibles (pixels éteints).

3 (c) couleur
choix d'une nouvelle couleur (code des 16 couleurs de 0 à F).

4 (f) fond
l'écran est colorié en couleur de fond (pixels éteints).

5 (r) remplissage
la surface intérieure à la ligne fermée est coloriée (pixels allumés) avec choix de couleur code 0 à F).

3 : Droite

1 (a) axes
trace deux axes perpendiculaires (pixels allumés).

2 (s) segment
place une mire fixe en extrémité d'un segment (1er choix), puis trace un segment jusqu'à la mire mobile (2ème choix).

3 (h) horizontale
trace une droite horizontale.

4 (v) verticale
trace une droite verticale.

(t)
retour au menu principal.

4 : curviligne

remplissage de surface autour d'un point pivot.

1 (a)
en allumant les pixels.

2 (e)
en coloriant le fond (pixels éteints).

4 (s)
confirmation ou annulation de l'option remplissage curviligne.

5 (p)
mise en place du point pivot avec choix de la couleur (code 0 à F).

(t)
retour au menu principal.

5 : écran

nettoyage de l'écran (pixels éteints).

Les options *visible, invisible, fond et curviligne* s'annulent en les choisissant deux fois. La mire se déplace au crayon optique ou avec les touches *W, E, R, S, D, Z, X et C.*

Liste des composants nécessaires

1 Photo-transistor Texas type TIL81
2 amplis opérationnels LF156 (circuits intégrés, boîtier 8 broches)
4 résistances 82KΩ, 1/4W
1 résistance 68KΩ, 1/4W
1 résistance 2,2KΩ, 1/4W
1 condensateur 47nF, 250V
1 condensateur 82nF, 250V
1 connecteur CANON 9 broches femelle avec capot
1,5 mètre de câble 4 conducteurs blindés
1 plaquette d'essai à bandes, de 1 cm * 10 cm (4 rangées de 39 trous). Les bandes cuivrées doivent être perpendiculaires à la grande longueur de la plaquette.
1 tube plastique de 10mm de diamètre intérieur et de 16 cm de longueur (glisser la plaquette cablée à l'intérieur du tube, montage légèrement serré)
2 embouts plastiques, un côté cellule photo transistor, un côté sortie câble vers console. Fermer le tout en collant au trichlo
2 portes fusibles à boulonner
2 fusibles de 300mA

Les connexions sur la plaquette sont à faire en fil souple, fin et isolés (type téléphone), environ 2 mètres.

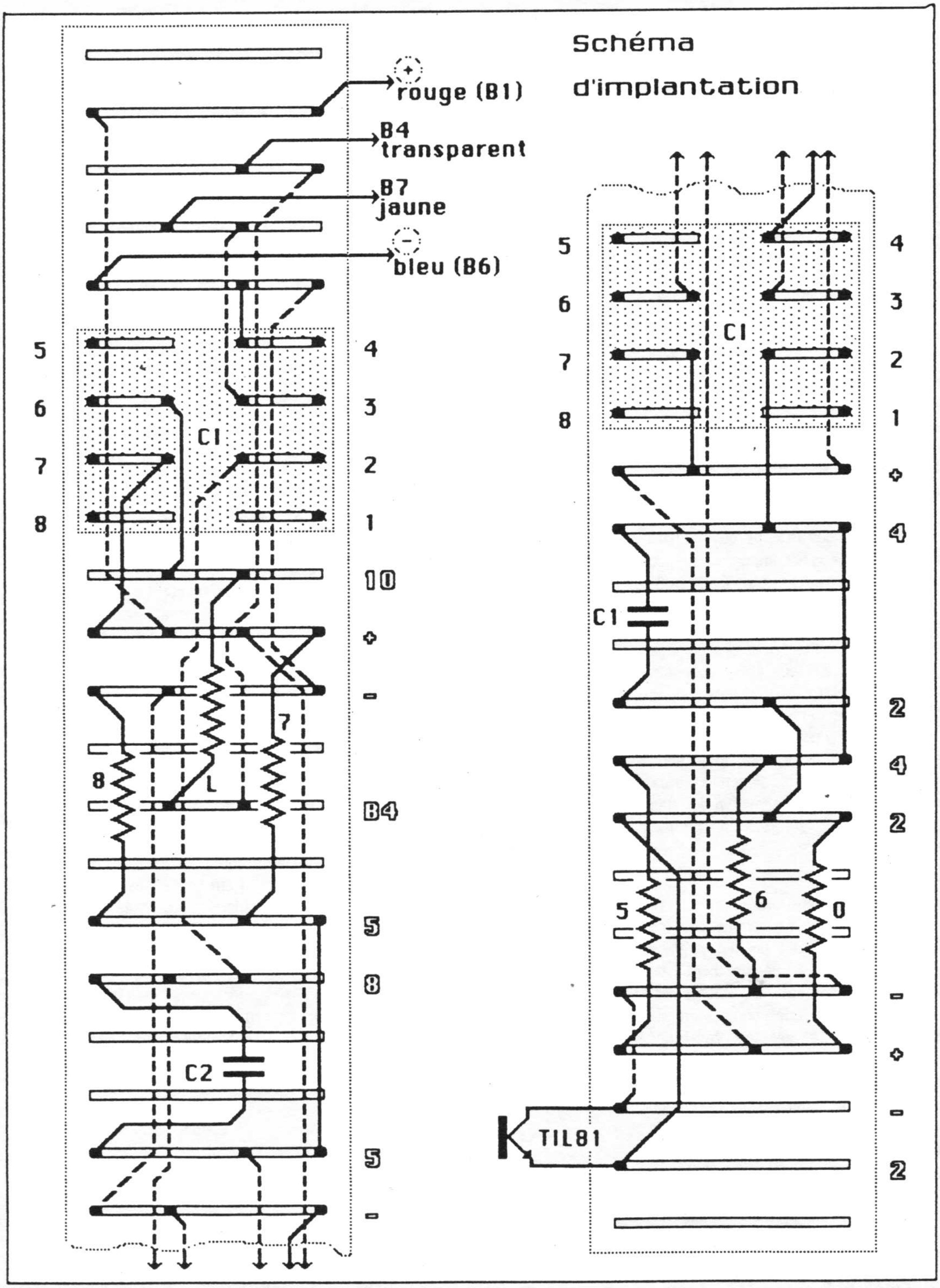
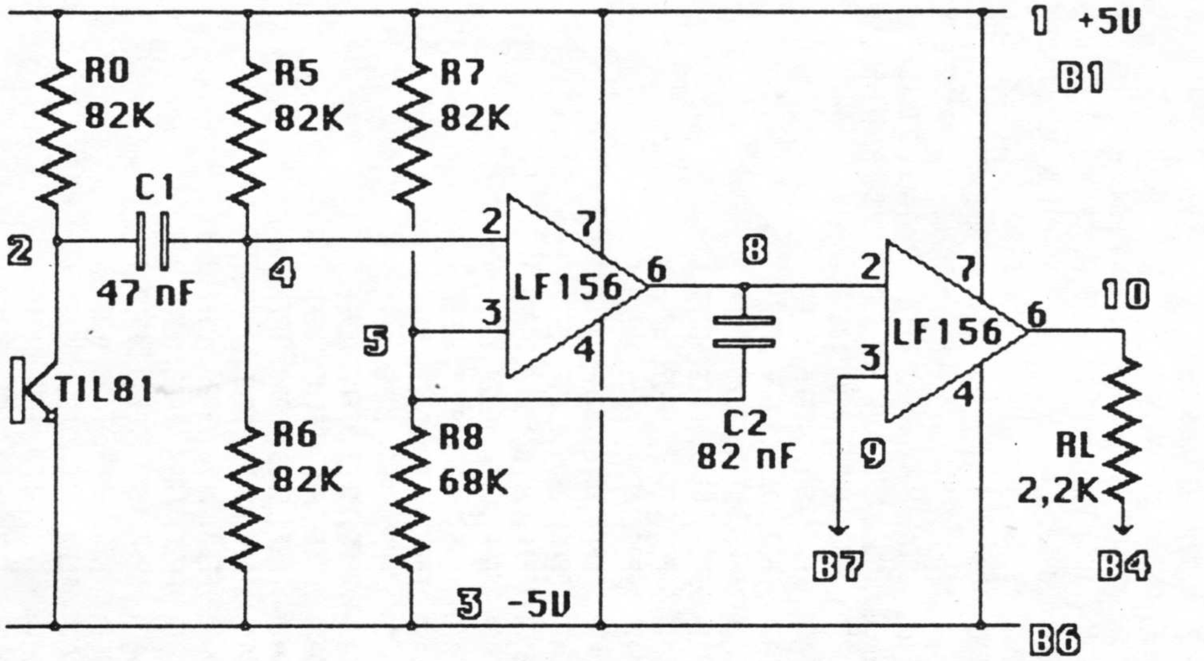
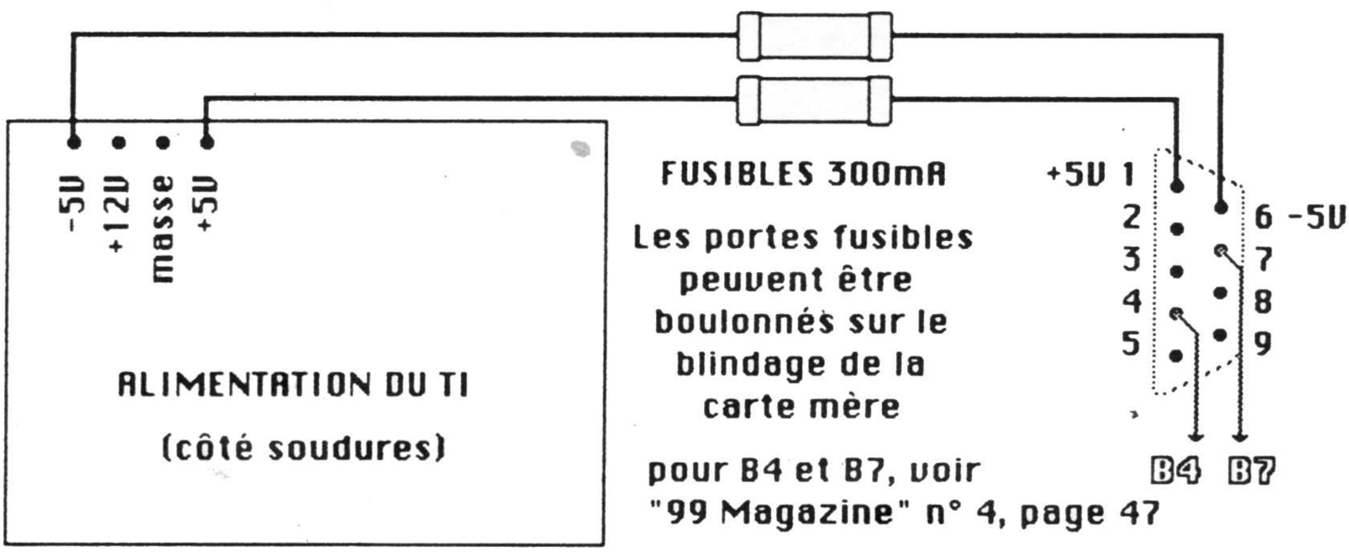

Schéma
d'implantation
rouge (B1)
B4
transparent
B7
jaune
bleu (B6)
C1
5
6
7
8
4
3
2
1
10
+
-
8
L
7
B4
5
8
C2
5
-
C1
+
4
C1
2
4
2
5
6
0
-
+
-
TIL81
2

Schéma théorique





Alimentation du crayon optique

Source COPY/S

```
COPY "DSK1.REFE/S"
COPY "DSK1.REMP/S"
COPY "DSK1.CRAY/S"
COPY "DSK1.STOR/S"
COPY "DSK1.MENU/S"
COPY "DSK1.NEWC/S"
COPY "DSK1.GRAF/S"
COPY "DSK1.PROG/S"
```

Source REFE/S

```
*********************************
*                               *
* PROGRAMME DE DESSIN A L'ECRAN *
*                               *
*    AVEC LE CRAYON OPTIQUE     *
*                               *
*       OU LE CLAVIER           *
*                               *
*********************************
*                               *
*   VERSION EDITOR/ASSEMBLER    *
*                               *
*********************************
*                               *
*    COPYRIGHT  99 MAGAZINE     *
*                               *
*              &                *
*                               *
*         BERNARD BEZUEL        *
*                               *
*********************************
*                               *
* PROGRAMME DE LANCEMENT:CRAYON *
* EXECUTION AVEC L'OPTION BASIC *
* DISQUETTE DANS LECTEUR DSK1   *
*                               *
*********************************
*
** REFE/S ** REFERENCES/DATA **
       DEF  CRAYON
       REF  VSBW,VMBW,VSBR,VMBR
       REF  DSRLNK,KSCAN,VWTR
       AORG >2700                  TAMPON ASCII (512 ADRESSES MEMOIRE)
PAB    EQU  >1F00                  BLOCK POUR ACCES DISQUE
PNTR   EQU  >8356                  POINTEUR POUR FICHIER SUR DISQUE
RTCLAV EQU  >8375                  RETOUR CLAVIER
STATUS EQU  >837C                  BYTE D'ETAT
TABLUT EQU  >1B80                  TABLE D'ATTRIBUTS DES LUTINS
USRWS  EQU  >20BA                  WORSPACE UTILISATEUR
*CARACTERES MINUSCULES NOIR SUR BLANC (CODE 32 A 95)
TALPHA DATA >0000,>0000,>0000,>0000 espace
       DATA >FFEF,>EFEF,>EFFF,>EFFF !
       DATA >FFD7,>D7D7,>FFFF,>FFFF "
       DATA >FFFF,>D783,>D783,>D7FF #
       DATA >FFEF,>C3AF,>C7EB,>87FF $
       DATA >FF9F,>9BF7,>EFD9,>B9FF %
       DATA >FFDF,>AFAF,>DFAB,>B7CB &
       DATA >FFFF,>FBFB,>F7FF,>FFFF '
       DATA >FFFF,>EFDF,>DFDF,>EFFF (
       DATA >FFFF,>F7FB,>FBFB,>F7FF )
       DATA >FFFF,>D7EF,>83EF,>D7FF *
       DATA >FFFF,>EFEF,>83EF,>EFFF +
       DATA >FFFF,>FFFF,>CFCF,>EFDF ,
       DATA >FFFF,>FFFF,>87FF,>FFFF -
       DATA >FFFF,>FFFF,>FFCF,>CFFF .
       DATA >FFFF,>FBF7,>EFDF,>BFFF /
       DATA >FFE7,>DBDB,>DBDB,>E7FF 0
       DATA >FFE7,>D7F7,>F7F7,>E3FF 1
       DATA >FFE7,>DBFB,>F7EF,>C3FF 2
       DATA >FFE3,>FBF7,>FBDB,>E7FF 3
       DATA >FFF7,>E7D7,>D7C3,>F7FF 4
       DATA >FFE3,>EFE7,>FBDB,>E7FF 5
       DATA >FFE7,>DFC7,>DBDB,>E7FF 6
       DATA >FFC3,>FBF7,>EFEF,>EFFF 7
       DATA >FFE7,>DBE7,>DBDB,>E7FF 8
       DATA >FFE7,>DBE3,>FBDB,>E7FF 9
       DATA >FFFF,>FFEF,>FFFF,>EFFF :
       DATA >FFFF,>FFEF,>FFCF,>EFDF ;
```

```
       DATA >FFFF,>F7EF,>DFEF,>F7FF <
       DATA >FFFF,>FFFF,>C3FF,>C3FF =
       DATA >FFFF,>DFEF,>F7EF,>DFFF >
       DATA >FFCF,>B7EF,>EFFF,>EFFF ?
       DATA >FFFF,>E3DB,>D3DF,>E3FF @
       DATA >FFFF,>C7B7,>B7B7,>CBFF a
       DATA >9FDF,>C7DB,>DBDB,>87FF b
       DATA >FFFF,>C7BB,>BFBB,>C7FF c
       DATA >F3F7,>C7B7,>B7B7,>C3FF d
       DATA >FFFF,>C7BB,>83BF,>C7FF e
       DATA >E7DB,>DF8F,>DFDF,>DFFF f
       DATA >FFE3,>DBDB,>E3FB,>DBE7 g
       DATA >9FDF,>D7CB,>DBDB,>DBFF h
       DATA >EFFF,>8FEF,>EFEF,>83FF i
       DATA >F7FF,>E7F7,>F7F7,>B7CF j
       DATA >DFDF,>DBD7,>CFD7,>DBFF k
       DATA >CFEF,>EFEF,>EFEF,>83FF l
       DATA >FFFF,>57AB,>ABAB,>ABFF m
       DATA >FFFF,>A7DB,>DBDB,>DBFF n
       DATA >FFFF,>C7BB,>BBBB,>C7FF o
       DATA >FFFF,>87DB,>DBC7,>DF8F p
       DATA >FFFF,>C3B7,>B7C7,>F7E3 q
       DATA >FFFF,>A7DB,>DFDF,>DFFF r
       DATA >FFFF,>C7BF,>C7FB,>87FF s
       DATA >DFDF,>87DF,>DFDB,>E7FF t
       DATA >FFFF,>B7B7,>B7B7,>CFFF u
       DATA >FFFF,>BBBB,>D7D7,>EFFF v
       DATA >FFFF,>95D5,>D5D5,>EBFF W
       DATA >FFFF,>BBD7,>EFD7,>BBFF X
       DATA >FFFF,>DBDB,>DBE3,>FBC7 Y
       DATA >FFFF,>83B7,>EFDB,>83FF Z
       DATA >FFFF,>CFDF,>DFDF,>CFFF [
       DATA >FFFF,>FFDF,>EFF7,>FBFF \
       DATA >FFFF,>E7F7,>F7F7,>E7FF ]
       DATA >FFEF,>D7BB,>FFFF,>FFFF ^
       DATA >FFFF,>FFFF,>FFFF,>C3FF _
*BUFFER/DATA DIVERS
GRAFWS BSS  >20  *ADRESSE >2900*     ESPACE DE TRAVAIL ROUTINES GRAPHIQUES
SVCLAV DATA 0                        SAUVEGARDE R0 (RETOUR CLAVIER)
SVY    DATA 0                        SAUVEGARDE Y
SVX    DATA 0                        SAUVEGARDE X

SVCOL  DATA 0                        SAUVEGARDE COULEUR (MENU)
COLORS DATA >2000                    COULEUR INITIALE
TABCOL DATA >2100,>3100,>4100,>5100  PALETTE COULEURS
       DATA >6100,>7100,>8100,>9100
       DATA >A100,>B100,>C100,>D100
       DATA >E100,>F100,>0100,>1100
MENU   DATA MENU10                   PREMIER MENU AFFICHE
MASQUE DATA MENU00
MENU00 BYTE 32,32,32,32,32,32        LIGNE 24 VIDE
       BYTE 32,32,32,32,32,32
       BYTE 32,32,32,32,32,32
       BYTE 32,32,32,32,32,32
       BYTE 32,32,32,32,32,32,32,32
XMENU  BYTE >08,>38,>68,>98,>C8,>F8  ABSCISSE LUTINS MASQUE OPTION
YMENU  BYTE >B7                      ORDONNEE LUTINS MASQUE OPTION
YLUTIN BYTE >AE,>AA,>A5,>A5,>A5,>AA,>AE,>AE,>AE,>C0 ORDONNEES POUR LUTINS
OUTECR DATA >C000                    ORDONNEE POUR EFFACEMENT LUTIN
SENSXY DATA >2220,>2101,>1110,>1202  DIRECTION DE DEPLACEMENT SUR L'ECRAN
LIGNES DATA 0,0                      2 POINTS POUR SEGMENT
COORD  DATA >AE1B,>0000              COORDONNEES DU POINT SUR ECRAN
ATRIBU DATA >C01C,>000F              LUTIN 1
       DATA >C01C,>010F                "   2
       DATA >C01C,>020F                "   3
       DATA >C018,>030F                "   4
       DATA >C013,>040F                "   5
       DATA >C013,>050F                "   6
       DATA >C013,>060F                "   7
       DATA >C018,>070F                "   8
       DATA >C018,>0A01              MASQUE
       DATA >C000,>080F              MIRE SEGMENT
       DATA >AA18,>0808              MIRE
       DATA >AA18,>090F              FLASH
       DATA >C008,>0B0F              MENU ALLUME
       DATA >C010,>0B0F              MENU ALLUME
       DATA >C018,>0B0F              MENU ALLUME
       DATA >C020,>0B0F,>D000        MENU ALLUME
LUTINS DATA >2030,>7C7F,>7F3F,>3F1F  LUTIN 1 CODE >0
       DATA >030F,>3F3F,>1F0F,>0300    "   2  "    >1
       DATA >1F3F,>3F7F,>7FFC,>3000    "   3  "    >2
       DATA >FEFC,>7C78,>3830,>0000    "   4  "    >3
       DATA >F8FC,>FCFE,>FE3E,>0C04    "   5  "    >4
```

```
        DATA >C0F0,>F8FC,>FCF0,>C000    "  6  "   >5
        DATA >000C,>3FFE,>FEFC,>FCF8    "  7  "   >6
        DATA >0000,>183B,>3C7C,>7EFE    "  8  "   >7
        DATA >1010,>00C6,>0010,>1000 MIRE     "   >8
        DATA >0000,>383B,>3800,>0000 FLASH    "   >9
        DATA >EEFE,>FE7C,>FEFE,>EE00 MASQUE   "   >A
        DATA >FFFF,>FFFF,>FFFF,>FFFF MASQUE MENU  >B
DP0     DATA 0                       DRAPEAU FLASH
DP1     DATA 0                       DRAPEAU TRACE CONTINU
DP2     DATA 0                       DRAPEAU TRACE DE SEGMENT
DP3     DATA 1                       DRAPEAU EFFACEMENT
DP4     DATA 0                       DRAPEAU REMPLISSAGE CURVILIGNE
DP5     DATA 0                       DRAPEAU AUXI. REMPLISSAGE CURVILILIGNE
DP6     DATA 0                       DRAPEAU REMPLISSAGE COULEUR DE FOND
DP7     DATA 0                       DRAPEAU REMPLISSAGE SURFACE HORIZONTALE
DP8     DATA 0                       DRAPEAU ECRITURE/LECTURE SUR DISQUE
DP9     DATA 0                       DRAPEAU RESERVE
P1      DATA 1
M1      DATA -1
```

Source REMP/S

```
***  REMPLISSAGE SURFACE  ***
RTREMP DATA 0   *
RTBOUT DATA 0    *
RT1LIN DATA 0     * SAUVEGARDE RETOUR APPEL (R11)
RTSECT DATA 0    *
RTBIT  DATA 0   *
HB     DATA 0  SENS REMPLISSAGE VERTICAL  (-1 HAUT, 1 BAS)
GD     DATA 0  SENS REMPLISSAGE HORIZONTAL (-1 GAUCHE, 1 DROITE)
DAV    DATA 0  ABSCISSE EXTREMITE DROITE AVANT-DERNIERE LIGNE
GAV    DATA 0  ABSCISSE EXTREMITE GAUCHE AVANT-DERNIERE LIGNE
GAP    DATA 0  ABSCISSE EXTREMITE GAUCHE DERNIERE LIGNE
AR     DATA 0,0 *
       DATA 0,0  *
       DATA 0,0  *
       DATA 0,0  *
       DATA 0,0  *
       DATA 0,0  *
       DATA 0,0  *
       DATA 0,0   * DONNEES DES POINTS D'ARRET
       DATA 0,0   * Y/X,HB (X=POINT DROIT DU SEGMENT)
       DATA 0,0  *
       DATA 0,0  *
       DATA 0,0  *
       DATA 0,0  *
       DATA 0,0  *
       DATA 0,0  *
       DATA 0,0 *
REMPLI MOV  R11,@RTREMP
       LWPI GRAFWS
       MOV  @COORD,@AR  Y/X DANS ARRO
       AB   @P1+1,@AR   Y+1
       MOV  @P1,@AR+2   HB=1
       LI   R3,4        INCREMENT STOCKAGE POINTS D'ARRET
RE105  AI   R3,-4       DEBUT DU REMPLISSAGE D'UN SECTEUR
       JLT  RE150       PLUS DE SECTEUR A REMPLIR
       CLR  R0
       MOVB @AR+1(R3),R0  ABSCISSE BORNE DROITE
       SWPB R0
       CLR  R1
       MOVB @AR(R3),R1  ORDONNEE BORNE DROITE
       SWPB R1
       MOV  @AR+2(R3),@HB  SENS DU REMPLISSAGE VERTICAL
       MOV  R0,@GAV     EXTREMITE GAUCHE SEGMENT PRECEDENT
       MOV  R0,@DAV     EXTREMITE DROITE SEGMENT PRECEDENT
*CONTROLE MIRE SUR POINT DEJA ALLUME
       MOV  R0,R10
       BL   @REBIT
       JEQ  RE150
RE110  BL   @RE1LIN    TRACE D'UN SEGMENT
       CI   R1,4       CONTROLE ECRAN HAUT
       JLT  RE105
       CI   R1,180     CONTROLE ECRAN BAS
       JGT  RE105
       MOV  @M1,@GD    CONTROLE CONTINUITE A GAUCHE AVEC SEGMENT PRECEDENT
       C    @GAV,@GAP MEME EXTREMITE GAUCHE
       JEQ  RE120
       C    @GAV,@GAP
       JGT  RE115      115->NOUVELLE EXTREMITE A GAUCHE DE L'ANCIENNE
       MOV  @GAP,R10  NOUVELLE EXTREMITE A DROITE DE L'ANCIENNE
```

```
        MOV  @GAV,R12  R10:POINT DE RECHERCHE;R12:BUTEE DE RECHERCHE
        DEC  R12
        BL   @RESECT   NOUVEAUX SECTEURS A CREER ?
        JMP  RE120     POUR RECHERCHE A DROITE
RE115   MOV  @GAV,R10  NOUVELLE EXTREMITE A GAUCHE DE L'ANCIENNE
        MOV  @GAP,R12  R10:POINT DE RECHERCHE;R12:BUTEE DE RECHERCHE
        NEG  @HB       REMPLISSAGE EN SENS INVERSE POUR NOUVEAUX SECTEURS
        A    @HB,R1    RETOUR POUR RECHERCHE SUR SEGMENT PRECEDENT
        DEC  R12
        BL   @RESECT   NOUVEAUX SECTEURS A CREER ?
        NEG  @HB       REPRENDRE ANCIEN SENS DE REMPLISSAGE
        A    @HB,R1    RETOUR SUR DERNIER SEGMENT
RE120   MOV  @GAP,@GAV FIN DU CONTROLE A GAUCHE->NOUVELLE BORNE GAUCHE GAV
        A    @HB,R1    CALCULS BORNE,CONTINUITE DROITE AVEC SEGME. PRECED.
        MOV  @DAV,R10  R10:POINT DE RECHERCHE
        BL   @REBIT    PIXEL ALLUME ?
        JEQ  RE135     135:PIXEL ALLUME,CHERCHER BORNE VERS LA GAUCHE
        MOV  @P1,@GD   DEBUT RECHERCHE BORNE VERS LA DROITE
        LI   R12,253   BORNE EXTREME DROITE
        NEG  @HB       REMPLISSAGE INVERSE POUR NOUVEAUX SECTEURS
RE125   A    @GD,R10
        C    R10,R12
        JEQ  RE130     BORNE DROITE ATTEINTE ?
        BL   @REBIT    PIXEL ALLUME ?
        JNE  RE125     125:CONTINUER LA RECHERCHE
RE130   MOV  @DAV,R12  BORNE GAUCHE POUR RECHERCHE EN SENS INVERSE
        DEC  R10       (SUITE)SUR LE SEGMENT PRECEDENT
        MOV  R10,@DAV  BORNE DROITE POUR SEGMENT SUIVANT
        INC  R10
        A    @HB,R1    REPASSER SUR SEGMENT PRECEDENT:RECHERCHE CONTINUITE
        NEG  @GD       RECHERCHE VERS LA GAUCHE
        BL   @RESECT   NOUVEAUX SECTEURS ?
        NEG  @HB       REPRENDRE SENS REMPLISSAGE PRECEDENT
        A    @HB,R1    RETOURNER SUR SEGMENT PRECEDENT
        JMP  RE110     FIN DES CALCULS,REMPLIR LE SEGMENT SUIVANT
RE135   MOV  @M1,@GD   DEBUT DE RECHERCHE BORNE VERS LA GAUCHE
        LI   R12,13    BORNE EXTREME GAUCHE
RE140   A    @GD,R10
        C    R10,R12   BORNE GAUCHE ATTEINTE ?
        JEQ  RE145     145:EXTREMITE GAUCHE SEGME. SUR BORD GAUCHE ECRAN
        BL   @REBIT    PIXEL ALLUME ?
        JEQ  RE140     140:CONTINUER LA RECHERCHE
RE145   C    R10,@GAV  BORNE A GAUCHE EXTREMITE GAUCHE SECTEUR PRECEDENT ?
        JLT  RE105     SI OUI:SECTEUR FERME->PASSER AU SECTEUR SUIVANT
        MOV  R10,@DAV  DAV:NOUVELLE BORNE DROITE POUR SEGMENT SUIVANT
        JMP  RE110     FIN DES CALCULS,REMPLIR LE SEGMENT SUIVANT
RE150   LWPI USRWS     FIN DU PROGRAMME
        MOV  @RTREMP,R11
        B    *R11
*** SOUS-PROGRAMMES ***
*TRACE D'UN SEGMENT HORIZONTAL ENTRE DAV ET PIXEL ALLUME A GAUCHE DE DAV
RE1LIN  MOV  R11,@RT1LIN
        MOV  @DAV,R0   BORNE DROITE
        MOV  R0,R2
        MOV  R0,R10
        BL   @REBIT    ALLUMER DANS R4 BIT CORRESPONDANT A DAV
        BL   @EXTREM   REMPLISSAGE EXTREMITE DROITE (BITS A GAUCHE DE DAV)
        CI   R4,>0100  OCTET CORRESPONDANT A R7 PLEIN A GAUCHE ?
        JNE  RE165     165:BORNE GAUCHE ATTEINTE,PASSER AU SEGMENT SUIVANT
        AI   R7,-8     8 PIXELS A GAUCHE
RE155   BL   @REVSBR   LECTURE NOUVEL OCTET R7 (8 PIXELS A GAUCHE)
        CI   R2,16     A GAUCHE DE L'ECRAN ?
        JLT  RE160     160 POUR REMPLISSAGE A GAUCHE
        MOV  R9,R9     AUCUN PIXEL ALLUME DANS LES 8 ?
        JNE  RE160     160 POUR REMPLISSAGE A GAUCHE
        LI   R10,>FF00 OCTET R7 VIDE:ALLUMER LES 8 PIXELS
        BL   @REVSBW   ALLUMER LES 8 PIXELS
        MOVB @COLORS,R10
        AI   R7,>2000  TABLE DES COULEURS
        BL   @REVSBW   COLORATION DES 8 PIXELS
        AI   R7,>E000  RETOUR A LA TABLE DES PATRONS
        AI   R7,-8     8 PIXELS A GAUCHE
        AI   R2,-8     8 PIXELS A GAUCHE
        JMP  RE155     POURSUITE REMPLISSAGE SEGMENT
RE160   LI   R4,>0001  PIXEL DROIT ALLUME DANS OCTET CORRESPOND A R7
        BL   @EXTREM   REMPLISSAGE EXTREMITE GAUCHE SEGMENT
RE165   INC  R2
        MOV  R2,@GAP   NOUVELLE BORNE GAUCHE POUR SEGMENT SUIVANT
        MOV  @RT1LIN,R11
        B    *R11
*ALLUMAGE BIT DANS R4 CORRESPONDANT A POSITION R10
REBIT   MOV  R11,@RTBIT
```

```
       MOV  R10,R0    CALCUL DU NUMERO DE L'OCTET CORRESPONDANT A R10
       MOV  R1,R7
       SLA  R7,5
       SOC  R1,R7
       ANDI R7,>FF07
       MOV  R0,R4
       ANDI R4,7
       A    R0,R7
       S    R4,R7     R7:NUMERO DE L'OCTET CORRESPONDANT A POSITION R10
       BL   @REVSBR   CALCUL DE R9 (PIXELS ALLUMES DANS R7/R10)
       LI   R4,>0080  BIT FORT ALLUME
       ANDI R0,7      R0:POSITION DU BIT DANS L'OCTET CORRESPONDANT A R10
       JEQ  RE170     R10 A GAUCHE DE L'OCTET:R4=>0080 (PIXEL GAUCHE)
       SRA  R4,0      DECALAGE A DROITE EGAL AUX 4 BITS FAIBLES DE R0
RE170  MOV  @RTBIT,R11
       SZC  R9,R4     BIT DE R4 SUR PIXEL ALLUME (DANS R9) ?
       B    *R11
*AFFICHAGE EXTREMITE DROITE A PARTIR DU BIT ALLUME DANS R4
EXTREM SZC  R9,R4     BIT R4 SUR PIXEL ALLUME ?
       JEQ  RE180     SI OUI AFFICHER L'OCTET R9
       SOC  R4,R9     SI NON ALLUMER DANS R9 LE BIT CORRESPONDANT A R4
       DEC  R2        1 PIXEL A GAUCHE
       CI   R2,12     BORD GAUCHE DE L'ECRAN ?
       JGT  RE175     SI NON DECALER BIT R4 D'UN PIXEL A GAUCHE
       INC  R2        SI OUI AFFICHER L'OCTET R9
       JMP  RE180
RE175  SLA  R4,1      1 PIXEL A GAUCHE
       CI   R4,>0100  PASSAGE DANS L'OCTET A GAUCHE ?
       JEQ  RE180     SI OUI L'OCTET EST ENTIEREMENT EXPLORE,AFFICHER R9
       JMP  EXTREM     SI NON DECALER BIT R4 D'UN PIXEL A GAUCHE
RE180  MOV  R11,@RTBOUT *
       MOV  R9,R10      *PREPARATION A L'AFFICHAGE
       SWPB R10         *
       BL   @REVSBW    ALLUMAGE BITS DE R9
       JMP  RE185      185 POUR COLORATION
RE1COL MOV  R11,@RTBOUT ENTREE POUR MODIF. COULEUR PIXEL DEJA ALLUME
RE185  AI   R7,>2000   TABLE DES COULEURS
       SWPB R7
       MOVB R7,@>8C02
       SWPB R7
       MOVB R7,@>8C02
       MOVB @COLORS,R10 COULEUR DANS R10
       MOVB @>8800,R13
       ANDI R10,>F000
       ANDI R13,>0F00
       SOCB R13,R10
       BL   @REVSBW    COLORATION DES PIXELS
       AI   R7,>E000   RETOUR A TABLE DES PATRONS
       MOV  @RTBOUT,R11
       B    *R11
*RECHERCHE SECTEURS ENTRE R10 ET BORNE R12
RESECT MOV  R11,@RTSECT
RE190  A    @GD,R10    DECALAGE 1 PIXEL VERS R12
       C    R10,R12    BORNE R12 ATTEINTE ?
       JEQ  RE200      SI OUI FIN DE LA RECHERCHE
       BL   @REBIT     RECHERCHE PIXEL ETEINT (DISCONTINUITE)
       JEQ  RE190      PIXEL ALLUME,CONTINUER LA RECHERCHE
       MOV  R10,@AR(R3) PIXEL ETEINT->OUVRIR UN NOUVEAU SECTEUR
       SWPB R1         R10 DANS AR+1:ABSCISSE DROITE DU SEGMENT
       MOVB R1,@AR(R3) R1 DANS AR:ORDONNEE DU SEGMENT
       SWPB R1
       MOV  @HB,@AR+2(R3) HB DANS AR+2:SENS DE REMPLISSAGE DU SECTEUR
       AI   R3,4       1 SECTEUR SUPPLEMENTAIRE A REMPLIR
RE195  A    @GD,R10    RECHERCHE FIN DU NOUVEAU SECTEUR
       C    R10,R12    BORNE ATTEINTE ?
       JEQ  RE200      SI OUI FIN DE LA RECHERCHE
       BL   @REBIT     PIXEL ALLUME ? (FIN DU NOUVEAU SECTEUR)
       JNE  RE195      SI NON CONTINUER RECHERCHE FIN DE SECTEUR
       JMP  RE190      FIN DU SECTEUR -> RECHERCHE AUTRE SECTEUR
RE200  MOV  @RTSECT,R11
       B    *R11
** LECTURE BYTE VDP **
REVSBR SWPB R7
       MOVB R7,@>8C02
       SWPB R7
       MOVB R7,@>8C02
       CLR  R9
       MOVB @>8800,R9
       SWPB R9
       B    *R11
** ECRITURE BYTE VDP **
REVSBW MOV  R7,R8
```

```
       LI   R12,>4000
       SWPB R8
       MOVB R8,@>8C02
       SWPB R8
       SOC  R12,R8
       MOVB R8,@>8C02
       NOP
       MOVB R10,@>8C00
       B    *R11
```

Source CRAY/S

```
*TEST SUR LUTINS 1 A 8 RECHERCHE POSITION CRAYON (X/Y)
RTCRA1 DATA 0           RETOUR A L'APPEL
RTCRA2 DATA 0           RETOUR A L'APPEL
CRAY01 MOV  R11,@RTCRA1 POUR RETOUR PROGRAMME PRINCIPAL
       MOV  @SVCLAV,R4
       CI   R4,>1200    CRAYON ECLAIRE ?
       JEQ  CR90
       CLR  R9          DRAPEAU REPETITION
CR05   MOVB @YLUTIN+5,@ATRIBU+40
       MOVB @ATRIBU+13,@ATRIBU+41
       LI   R0,TABLUT+40
       LI   R1,ATRIBU+40
       LI   R2,4
       BLWP @VMBW
       LI   R8,2000     TEMPO NOUVELLE TOUCHE
       LI   R7,600      TEMPO PREMIERE TOUCHE
CR10   BLWP @KSCAN
       MOVB @STATUS,@STATUS
       JNE  CR35        NOUVELLE TOUCHE TRAITEE DE SUITE(SANS REPETITION)
CR15   MOVB @RTCLAV,R4
       CI   R4,>FF00    AUCUNE TOUCHE ?
       JEQ  CR30        VERS TEMPO ATTENTE TOUCHE
       CI   R9,3        REPETITION ?
       JLT  CR25        PAS DE REPETITION
       LI   R7,200      VITESSE REPETITION
CR20   DEC  R7
       JNE  CR20
       JMP  CR40
CR25   DEC  R7
       JNE  CR10        TEMPO PREMIERE TOUCHE
       INC  R9
       JMP  CR40
CR30   DEC  R8          TEMPO ATTENT NOUVELLE TOUCHE
       JNE  CR10
       B    @CR140      SORTIE
CR35   CLR  R9          REMISE A ZERO REPETITION
CR40   MOVB @RTCLAV,R4
       JNE  CR45        X(LUTIN8)
       LI   R3,7
       JMP  CR80
CR45   CI   R4,>0E00    C(LUTIN1)
       JNE  CR50
       CLR  R3
       JMP  CR80
CR50   CI   R4,>0300    D(LUTIN2)
       JNE  CR55
       LI   R3,1
       JMP  CR80
CR55   CI   R4,>0600    R(LUTIN3)
       JNE  CR60
       LI   R3,2
       JMP  CR80
CR60   CI   R4,>0500    E(LUTIN4)
       JNE  CR65
       LI   R3,3
       JMP  CR80
CR65   CI   R4,>0400    W(LUTIN5)
       JNE  CR70
       LI   R3,4
       JMP  CR80
CR70   CI   R4,>0200    S(LUTIN6)
       JNE  CR75
       LI   R3,5
       JMP  CR80
CR75   CI   R4,>0F00    Z(LUTIN7)
       JNE  CR85
       LI   R3,6
CR80   BL   @MODXY
CR85   JMP  CR05
```

```
*TRAITEMENT CRAYON
CR90   LI   R2,4      4 BYTES A DEPLACER VERS VDP(TABLE ATTRIBUT LUTIN)
       LI   R10,63    TEMPORISATION
       CLR  R12       DARR=0 SENS TEST VERS LUTIN SUIVANT
       LI   R13,4
       MOVB @YLUTIN+1,@ATRIBU+32 ORDONNEE POUR MASQUE SUR DESSIN
CR95   LI   R0,TABLUT ADRESSE VDP LUTIN 1
       LI   R1,ATRIBU ADRESSE LUTIN 1
       CLR  R3        POUR TEST SUR LUTIN 1
       MOV  R0,R7
       MOV  R1,R8
CR100  LI   R9,20     TEMPO. REPONSE CRAYON
CR105  LI   R0,TABLUT+32 ADRESSE MASQUE DANS VDP
       LI   R1,ATRIBU+32
       BLWP @VMBW     MASQUE SUR ECRAN
       MOV  R7,R0
       MOV  R8,R1
       MOVB @YLUTIN(R3),*R1 LUTIN (R3+1) SUR ECRAN
       BLWP @VMBW
       CLR  R4
CR110  BLWP @KSCAN
       MOVB @RTCLAV,R4
       CI   R4,>1200  CODE TIR (CRAYON ECLAIRE)
       JEQ  CR125     MODIFICATION XY
       DEC  R9
       JNE  CR110     BOUCLAGE POUR REPONSE CRAYON (KSCAN)
       MOV  R7,R0
       MOV  R8,R1
       MOVB @OUTECR,*R1 LUTIN TESTE MIS HORS ECRAN
       BLWP @VMBW
       DEC  R10
       JEQ  CR130     FIN TEMPO. PLACEMENT CRAYON SUR ECRAN
       MOV  R12,R12   SENS DES TEST ?(LUTIN SUIVANT OU PRECEDENT)
       JEQ  CR120     TEST SUR LUTIN SUIVANT
       CLR  R12       TEST SUR LUTIN PRECEDENT PUIS SUIVANT
       MOV  R3,R3     TEST SUR LUTIN 1 ?
       JEQ  CR115     TEST SUR LUTIN 8(PRECEDENT)
       DEC  R3        TEST SUR LUTIN PRECEDENT
       S    R13,R7    LUTIN PRECEDENT
       S    R13,R8    ATTRIBUT LUTIN PRECEDENT
       JMP  CR100     REPRISE CYCLE DES TESTS
CR115  LI   R0,TABLUT+28 ADRESSE LUTIN 8
       LI   R1,ATRIBU+28  ATTRIBUTS LUTIN 8
       LI   R3,7      TEST SUR LUTIN 8(R3+1)
       MOV  R0,R7
       MOV  R1,R8
       JMP  CR100     REPRISE CYCLE DES TESTS
CR120  CI   R3,7      TEST 8 EN COURS ?
       JEQ  CR95      VERS TEST 1
       INC  R3        TEST LUTIN SUIVANT
       A    R13,R7    LUTIN SUIVANT
       A    R13,R8    ATTRIBUT LUTIN SUIVANT
       JMP  CR100     REPRISE CYCLE DES TESTS
CR125  BL   @MODXY
       LI   R10,39    TEMPORISATION
       INC  R12       DARR=1 POUR FUTUR TEST SUR LUTIN SUIVANT
       LI   R9,50     TEMPO
       B    @CR105
CR130  LI   R0,>8000  TEMPO POUR RETIRER LE CRAYON DE L'ECRAN
CR135  DEC  R0
       JNE  CR135
       MOVB @OUTECR,@ATRIBU+32   ADRESSE MASQUE HORS ECRAN
       LI   R0,TABLUT+32
       LI   R1,ATRIBU+32
       BLWP @VMBW     MASQUE HORS ECRAN
CR140  MOV  @RTCRA1,R11
       B    *R11
*MODIFICATION POSITION X/Y CRAYON SUR ECRAN
MODXY  MOV  R11,@RTCRA2
       CLR  R0
       MOVB @SENSXY(R3),R0 CODE DEPLACEMENT DU LUTIN NUMERO R3+1
       MOV  R0,R4
       ANDI R4,>0200  DIRECTION VERS HAUT DE L'ECRAN ? (Y-1)
       JNE  CR145     SI NON TEST POUR Y+1
       LI   R6,YLUTIN ADRESSE ORDONNEE LUTIN 1
       LI   R1,-1     INCREMENT -1 POUR MODIF.Y
       LI   R4,>0300  LIMITE HAUTE DE L'ECRAN
       LI   R5,1      1 DANS R5 POUR MODIF.Y DES LUTINS (@Y+...)
       BL   @CR175
CR145  MOV  R0,R4
       ANDI R4,>0100  DIRECTION VERS BAS DE L'ECRAN ? (Y+1)
       JNE  CR150
```

```
       LI    R6,YLUTIN
       LI    R1,1       INCREMENT +1 POUR MODIF.Y
       LI    R4,>B500   LIMITE BASSE DE L'ECRAN
       LI    R5,1       1 DANS R5 POUR MODIF Y DES LUTINS (@Y+...)
       BL    @CR175
CR150  MOV   R0,R4
       ANDI  R4,>2000   DIRECTION VERS LA GAUCHE DE L'ECRAN ? (X-1)
       JNE   CR155
       LI    R6,COORD+1 ADRESSE ABSCISSES POINT ET LUTINS
       LI    R1,-1      INCREMENT -1 POUR MODIFICATION X
       LI    R4,>0D00   LIMITE GAUCHE DE L'ECRAN
       LI    R5,4       4 DANS R5 POUR MODIF.X DES LUTINS (@ATRIBU+...)
       BL    @CR175
CR155  MOV   R0,R4
       ANDI  R4,>1000   DIRECTION VERS LA DROITE DE L'ECRAN ? (X+1)
       JNE   CR160      POUR RETOUR APPEL
       LI    R6,COORD+1
       LI    R1,1       INCREMENT +1 POUR MODIF.X
       LI    R4,>FC00   LIMITE DROITE DE L'ECRAN
       LI    R5,4       4 DANS R5 POUR MODIF. X DES LUTINS (@ATRIBU+...)
       BL    @CR175
CR160  MOVB  @YLUTIN+8,@COORD NOUVELLE ORDONNEE Y DANS @COORD
       MOVB  @YLUTIN+1,@ATRIBU+32  NOUVELLE ORDONNEE POUR MASQUE
       MOVB  @ATRIBU+13,@ATRIBU+33 NOUVELLE ABSCISSE POUR MASQUE
       MOV   R0,@SVX
       MOV   R1,@SVY
       MOV   @DP1,@DP1  DRAPEAU 1 TRACE CONTINU
       JEQ   CR165      DRAPEAU 0 TRACE POINT PAR POINT
       BL    @POINT     TRACE CONTINU
       LWPI  USRWS
       JMP   CR170
CR165  MOV   @DP4,@DP4  DRAPEAU REMPLISSAGE CURVILIGNE
       JEQ   CR170      REMPLISSAGE CURVILIGNE
       INC   @DP2
       BL    @LIGNE
       CLR   @DP2
CR170  MOV   @RTCRA2,R11
       B     *R11
CR175  CB    *R6,R4     X OU Y SUR BORD DE L'ECRAN ?
       JEQ   CR185      SI OUI PAS DE MODIF. XY,RETOUR A L'APPEL
       LI    R4,9       SI NON,9 LUTINS A DEPLACER (INCREMENT 1)
       SWPB  R1
CR180  AB    R1,*R6     MODIF XY
       A     R5,R6      ADRESSE COORDONNEE SUIVANTE A MODIFIER
       DEC   R4         LUTIN SUIVANT
       JNE   CR180
CR185  B     *R11
```

Source STOR/S

```
MODE   DATA >0000     MODE LECTURE/ECRITURE
NOM    DATA 304,328,280,320,328,296,400,0 "FICHIER"*(8 CARACTERES)
       DATA 328,360,264,312,296,0,288,408,344 "IMAGE DSK"(9 CARACTERES)
PAS    DATA >80       NOMBRE DE BYTES PAR ENREGISTREMENT
PBUF   DATA 0         ADRESSE BUFFER VDP
PDATA  DATA >0000,>0000,>0001,PBUF,>8080,>0000,>000F
       TEXT 'DSKX.XXXXXXXXXX'
*TEST LECTURE/ECRITURE SUR DISQUE
STORE  MOV  @DP8,@DP8 DRAPEAU ECRITURE/LECTURE
       JEQ  LECTU
       LI   R0,>0300 MODE ECRITURE
       MOV  R0,@MODE
       JMP  ST05
LECTU  LI   R0,>0200 MODE LECTURE
       MOV  R0,@MODE
*DEMANDE NOM FICHIER
ST05   CLR  R3        NUMERO DE LA LETTRE A AFFICHER (DANS @NOM)
       LI   R0,>1708 DEUXIEME CASE LIGNE 24
       LI   R2,8      8 BYTES PAR CARACTERE
ST10   LI   R1,TALPHA DEBUT TAMPONS CARACTERES ASCII
       A    @NOM(R3),R1 ADRESSE DU CARACTERES A AFFICHER
       BLWP @VMBW
       AI   R0,8      CASE SUIVANTE SUR LIGNE 24
       INCT R3        LETTRE SUIVANTE
       CI   R3,34     FIN DU TAMPON @NOM ?
       JNE  ST10
       LI   R0,0      CLAVIER ENTIER
       MOVB R0,@>8374
*SAISIE NOM DU FICHIER IMAGE
ST15   LI   R2,8      8 BYTES A ECRIRE
       LI   R4,4      QUATRIEME CARACTERE DU NOM DU FICHIER (DSK.)
```

```
      LI   R0,>1790 CASE 19 LIGNE 24 (DISQUETTE 1 OU 2)
ST20  LI   R1,TALPHA+504 SOULIGNE
      BLWP @VMBW    AFFICHAGE
ST25  CLR  R3       VIDAGE R3 POUR STATUS (TOUCHE PRESSEE ?)
      BLWP @KSCAN   APPEL CLAVIER .
      MOVB @STATUS,R3 NOUVELLE TOUCHE PRESSEE ?
      JEQ  ST25     PAS DE TOUCHE PRESSEE
      MOVB @RTCLAV,R3 CODE TOUCHE PRESSEE (NUMERO DISK)
      CI   R4,4     SAISIE NUMERO DISK ?
      JNE  ST40     SAISIE NOM DE L'IMAGE
      CI   R3,>3100 CODE NOMBRE 1 ? (DSK1)
      JNE  ST30     POUR TEST SUR 2
      LI   R1,TALPHA+136 ADRESSE DU NOMBRE 1
      BLWP @VMBW    AFFICHAGE 1
      JMP  ST35     POUR TRANSFERT NUMERO DANS PDATA
ST30  CI   R3,>3200 CODE NOMBRE 2 ? (DSK2)
      JNE  ST25     RETOUR CLAVIER REFUSE (1 OU 2 VALABLE)
      LI   R1,TALPHA+144 ADRESSE DU NOMBRE 2
      BLWP @VMBW    AFFICHAGE 2
ST35  MOVB R3,@PDATA+17 NUMERO LECTEUR DISQUETTE DANS PDATA
      LI   R1,TALPHA+112 POINT
      AI   R0,8     CASE 20 LIGNE 24
      BLWP @VMBW    AFFICHAGE DU POINT
      LI   R4,6     SIXIEME LETTRE DU NOM DU FICHIER (DSKX.?)
      AI   R0,8     CASE 21 LIGNE 24
      JMP  ST20     APPEL PREMIER CARACTERE DU NOM DE L'IMAGE
ST40  CI   R3,>0800 TOUCHE RETOUR A GAUCHE ?
      JNE  ST45     TEST SUR CARACTERE
      CI   R4,6     PREMIERE LETTRE DU NOM DU FICHIER ?
      JEQ  ST15     REPREISE DES SAISIES AU DEBUT
      CI   R4,4     PREMIER CARACTERE ?
      JEQ  ST25     SOULIGNE DEJA A GAUCHE
      LI   R1,TALPHA CARACTERE ESPACE (EFFACEMENT SOULIGNE)
      BLWP @VMBW    EFFACEMENT SOULIGNE
      DEC  R4       CARACTERE PRECEDENT
      AI   R0,-8    CASE PRECEDENTE A L'ECRAN
      JMP  ST20     APPEL NOUVEAU CARACTERE DE REMPLACEMENT
ST45  CI   R4,6     TEST PREMIER CARACTERE (LETTRE ?)
      JNE  ST50     TEST VALIDITE CARACTERE (POINT/ESPACE REFUSE)
      CI   R3,>4100 LETTRE A
      JLT  ST25     PREMIER CARACTERE REFUSE
      CI   R3,>5A00 LETTRE Z
      JGT  ST25     PREMIER CARACTERE REFUSE
ST50  CI   R3,>2000 CARACTERE ESPACE INTERDIT
      JEQ  ST25     REFUSE (APPEL NOUVELLE TOUCHE)
      CI   R3,>2E00 CARACTERE POINT INTERDIT
      JEQ  ST25     REFUSE (APPEL NOUVELLE TOUCHE)
      CI   R3,>0D00 TOUCHE ENTER (FIN DU NOM)
      JEQ  ST55     FIN DE LA SAISIE DU NOM DU FICHIER
      MOVB R3,@PDATA+13(R4) CARACTERE DANS @PDATA
      LI   R1,TALPHA ADRESSE TAMPON CARACTERES ASCII
      SWPB R3       CODE ASCII DANS BYTE FAIBLE
      AI   R3,-32   CODE ASCII-32 (NUMERO D'ORDRE DANS TAMPON T)
      SLA  R3,3     8*R3 POUR ADRESSE DU CARACTERE DANS TAMPON T
      A    R3,R1    ADRESSE DU CARACTERE DANS TAMPON T
      BLWP @VMBW    AFFICHAGE DU CARACTERE
      CI   R4,15    DIXIEME CARACTERE ?
      JEQ  ST25     CORRECTION DERNIER CARACTERE ?
      INC  R4       CARACTERE SUIVANT
      AI   R0,8     CASE SUIVANTE A L'ECRAN (LIGNE 24)
      JMP  ST20     APPEL PROCHAIN CARACTERE
ST55  DEC  R4       TRANSFERT LONGUEUR DU NOM FICHIER DANS PDATA
      SWPB R4       VALEUR DANS BYTE FORT
      MOVB R4,@PDATA+13 LONGUEUR DU NOM DU FICHIER DANS @PDATA
      LI   R0,>0100 DEMI-CLAVIER GAUCHE
      MOVB R0,@>8374
*ENREGISTREMENT/LECTURE IMAGE SUR DISQUE
*OUVERTURE FICHIER
      LI   R0,PAB   PAB MODE OUVERTURE
      LI   R1,PDATA
      LI   R2,>20
      BLWP @VMBW    PAB DANS LA VDP
      LI   R6,PAB+13
      MOV  R6,@PNTR
      BLWP @DSRLNK  OUVERTURE DU FICHIER
      DATA 8
*ECRITURE/LECTURE SUR DISQUE
      MOVB @MODE,R1 PAB AU MODE ECITURE/LECTURE
      LI   R0,PAB+4
      BLWP @VSBW
      LI   R0,PAB+6
      CLR  @PBUF    ADRESSE DE LA TABLE DES PATRONS
```

```
       LI   R1,PBUF
       LI   R2,2
       BLWP @VMBW
       CLR  R3          NUMERO D'ENREGISTREMENT
ST60   MOV  R6,@PNTR
       BLWP @DSRLNK
       DATA 8
       INC  R3
       A    @PAS,@PBUF
       BLWP @VMBW
       CI   R3,46
       JNE  ST60
       LI   R1,>2000 ADRESSE DE LA TABLE DES COULEURS
       MOV  R1,@PBUF
       LI   R1,PBUF
       BLWP @VMBW
ST65   MOV  R6,@PNTR
       BLWP @DSRLNK
       DATA 8
       INC  R3
       A    @PAS,@PBUF
       BLWP @VMBW
       CI   R3,92
       JNE  ST65
*FERMETURE FICHIER
       LI   R1,>0100 PAB AU MODE FERMETURE
       LI   R0,PAB+4
       BLWP @VSBW
       MOV  R6,@PNTR
       BLWP @DSRLNK
       DATA 8           FERMETURE DU FICHIER
*EFFACEMENT LIGNE 24
       LI   R3,31       31 CASES A EFFACER
       LI   R0,>1708 PREMIERE CASE LIGNE 24
       LI   R1,TALPHA CARACTERE ESPACE
       LI   R2,8
ST70   BLWP @VMBW
       AI   R0,8        CASE SUIVANTE
       DEC  R3
       JNE  ST70        DERNIERE CASE ?
       B    *R11
```

Source MENU/S

```
$VMENU DATA 0  SAUVEGARDE MENU EN COURS
RTTM   DATA 0  RETOUR A L'APPEL
MENU10 BYTE 32,32,32,32,32,32
       BYTE 32,84,82,65,73,50          TRAI2
       BYTE 32,82,69,67,84,51          RECT3
       BYTE 32,67,85,82,86,52          CURV4
       BYTE 32,85,84,73,76,53,32,70 UTIL5 F(FIN D'EXECUTION)
       DATA MENU10,MEN110,MEN120,MEN130,MEN140,MENU10
       DATA RESE01,SAUT2,SAUT2,SAUT2,SAUT2,SAUT2
MEN100 BYTE 32,32,32,32,32,32
       BYTE 32,32,32,32,32,32
       BYTE 32,32,32,32,32,32
       BYTE 32,32,32,32,32,32
       BYTE 32,32,32,32,32,32,32,84         T(MENU)
       DATA MEN100,MEN100,MEN100,MEN100,MEN100,MENU10
       DATA RESE01,RESE01,RESE01,RESE01,RESE01,SAUT2
MEN110 BYTE 32,86,73,83,73,49          VISI1
       BYTE 32,69,70,70,65,50          EFFA2
       BYTE 32,67,79,85,76,51          COUL3
       BYTE 32,70,79,78,68,52          FOND4
       BYTE 32,84,89,80,69,53,32,84 TYPE5 T
       DATA MEN110,MEN110,MENU10,MEN110,MEN110,MENU10
       DATA RVISIB,REFFAC,RCOULE,RFOND,RESE01,SAUT2
MEN120 BYTE 32,65,88,69,83,49          AXES1
       BYTE 32,83,69,71,77,50          SEGM2
       BYTE 32,68,82,79,73,51          DROI3
       BYTE 32,82,69,76,73,52          RELI4
       BYTE 32,80,79,73,78,53,32,84 POIN5 T
       DATA MEN120,MEN120,MEN121,MEN120,MEN120,MENU10
       DATA RAXES,RLIGNE,SAUT2,RESE01,RPOINT,SAUT2
MEN121 BYTE 32,72,79,82,73,49          HORI1
       BYTE 32,86,69,82,84,50          VERT2
       BYTE 32,79,82,73,69,51          ORIE3
       BYTE 32,68,69,77,73,52          DEMI4
       BYTE 32,32,32,32,32,32,32,84         T
       DATA MEN121,MEN121,MEN121,MEN121,MEN121,MEN120
       DATA RHORIZ,RVERTI,RESE01,RESE01,RESE01,SAUT2
MEN130 BYTE 32,34,79,78,34,49          "ON"1
       BYTE 32,34,79,70,34,50          "OF"2
```

```
       BYTE 32,32,32,32,32,32
       BYTE 32,83,85,82,70,52        SURF4
       BYTE 32,80,73,86,79,53,32,84 PIVO5 T
       DATA MEN130,MEN130,MEN130,MEN130,MEN130,MENU10
       DATA RON,ROFF,RESE01,RCSURF,RCSUR2,SAUT2
MEN140 BYTE 32,67,65,68,82,49        CADR1
       BYTE 32,83,89,77,66,50        SYMB2
       BYTE 32,69,67,82,65,51        ECRA3
       BYTE 32,82,69,77,80,52        REMP4
       BYTE 32,83,85,73,84,53,32,84 SUIT5 T
       DATA MEN140,MEN140,MENU10,MEN140,MEN141,MENU10
       DATA RESE01,RESE01,RECRAN,RSURFA,SAUT2,SAUT2
MEN141 BYTE 32,79,70,83,69,49        OFSE1
       BYTE 32,90,79,79,77,50        ZOOM2
       BYTE 32,71,82,79,85,51        GROU3
       BYTE 32,70,69,78,69,52        FENE4
       BYTE 32,83,85,73,84,53,32,84 SUIT5 T
       DATA MEN141,MEN141,MEN141,MEN141,MEN142,MEN140
       DATA RESE01,RESE01,RESE01,RESE01,SAUT2,SAUT2
MEN142 BYTE 32,79,86,76,65,49        OVLA1
       BYTE 32,82,69,67,72,50        RECH2
       BYTE 32,69,78,82,69,51        ENRE3
       BYTE 32,32,32,32,32,32
       BYTE 32,32,32,32,32,32,32,84        T
       DATA MEN142,MENU10,MENU10,MEN142,MEN142,MEN141
       DATA RESE01,RSTORE,RSTORE,RESE01,RESE01,SAUT2
* PROGRAMME TEST MENU
TESMEN MOV  R11,@RTTM
       MOV  @MENU,@SVMENU SAUVEGARDE MENU EN COURS
       MOV  @MASQUE,@MENU
       BL   @AFFMEN        AFFICHAGE MASQUE DE LIGNE
       MOV  @SVMENU,@MENU RESTITUTION MENU EN COURS
       MOVB @YMENU,R4      *
       MOVB R4,@ATRIBU+48  *
       MOVB R4,@ATRIBU+52  *
       MOVB R4,@ATRIBU+56   * PREPARATION POUR AFFICHAGE LUTINS MENU
       MOVB R4,@ATRIBU+60  *
       LI   R0,TABLUT+48   *
       LI   R1,ATRIBU+48   *
       LI   R4,1000  TEMPO EXTINCTION CRAYON
ME05   DEC  R4
       JNE  ME05
       LI   R4,20     TEMPO ECLAIRAGE CRAYON
       CLR  R5
ME10   DEC  R4
       JEQ  ME15      POUR CONFIRMATION OPTION RETENUE
       BLWP @KSCAN
       MOVB @RTCLAV,R5 TOUCHE ENFONCEE OU CRAYON ECLAIRE
       CI   R5,>1200 POUR OPTION TESTEE
       JNE  ME10      CRAYON PAS ECLAIRE
       B    @ME80     CRAYON ECLAIRE->OPTION REFUSEE
ME15   CLR  R6        NUMERO OPTION
       CLR  R7
ME20   MOVB @XMENU(R6),R7 ABSCISSE OPTION (PREMIER LUTIN)
       MOVB R7,@ATRIBU+49 X DANS POSITION LUTIN1 MENU
       AI   R7,>0800      X+8 (LUTIN 2)
       MOVB R7,@ATRIBU+53
       AI   R7,>0800      X+16 (LUTIN3)
       MOVB R7,@ATRIBU+57
       AI   R7,>0800      X+24 (LUTIN 4)
       MOVB R7,@ATRIBU+61
       BLWP @VMBW         AFFICHAGE LUTINS (MASQUE OPTION)
ME25   LI   R4,24         TEMPO REPONSE CRAYON
ME30   BLWP @KSCAN
       MOVB @RTCLAV,R5
       CI   R5,>1200 CRAYON ECLAIRE ?
       JEQ  ME70      CONFIRMATION OPTION
       CI   R5,>1300 TOUCHE 1
       JNE  ME35
       LI   R6,0
       JMP  ME70
ME35   CI   R5,>0700 TOUCHE 2
       JNE  ME40
       LI   R6,1
       JMP  ME70
ME40   CI   R5,>0800 TOUCHE 3
       JNE  ME45
       LI   R6,2
       JMP  ME70
ME45   CI   R5,>0900 TOUCHE 4
       JNE  ME50
       LI   R6,3
```

```
       JMP  ME70
ME50   CI   R5,>0A00 TOUCHE 5
       JNE  ME55
       LI   R6,4
       JMP  ME70
ME55   CI   R5,>0B00 TOUCHE T
       JNE  ME60
       LI   R6,5
       JMP  ME70
ME60   DEC  R4         FIN DE TEMPO ?
       JNE  ME30
       INC  R6         OPTION SUIVANTE
       CI   R6,5       AVANT/DERNIERE OPTION ?
       JLT  ME20       OPTION SUIVANTE
       JEQ  ME65       PLUS D'OPTION
       BL   @MASQ      EFFACEMENT LIGNE 24
       B    @ME80      PLUS D'OPTION A TESTER
ME65   LI   R7,>F800 OPTION MENU (DERNIERE)
       MOVB R7,@ATRIBU+49 *
       MOVB R7,@ATRIBU+53 * 5 LUTINS SUR OPTION MENU (T)
       MOVB R7,@ATRIBU+57 *
       MOVB R7,@ATRIBU+61 *
       BLWP @VMBW
       JMP  ME25       DERNIER TEST OPTION M (MENU)
ME70   BL   @MASQ      EFFACEMENT MASQUE 4 CASES
       MOV  @SVMENU,R4 ADRESSE MENU EN COURS
       AI   R4,32      PREMIERE LIGNE DATA DANS TABLEAUX MENU
       SLA  R6,1       2*R6 (NUMERO D'ORDRE DU PROCHAIN MENU)
       A    R6,R4      ADRESSE DU PROCHAIN MENU DANS TABLEAUX
       MOV  *R4,@SVMENU SAUVEGARDE DU PROCHAIN MENU
       AI   R4,12      DEUXIEME LIGNE DATA DANS TABLEAUX MENU
       MOV  *R4,R1     ADRESSE DE LA ROUTINE A EXECUTER
       B    *R1        BRANCHEMENT POUR EXECUTION ROUTINE (DU MENU)
RPOINT BL   @POINT     ROUTINE POINT
       LWPI USRWS
       JMP  SAUT2
RAXES  BL   @HORIZ     ROUTINE HORIZONTALE
       BL   @VERTI     ROUTINE VERTICALE
       JMP  SAUT2
RLIGNE BL   @LIGNE     ROUTINE SEGMENT
       JMP  SAUT2
```

```
RSTORE CI   R6,2       LECTURE ? (OPTION MENU 2:R6=2)
       JEQ  RSTOR1
       INC  @DP8       DRAPEAU ECRITURE DANS FICHIER
       JMP  RSTOR2
RSTOR1 CLR  @DP8       DRAPEAU LECTURE DANS FICHIER
RSTOR2 BL   @STORE     ROUTINE LECTURE/ENREGISTREMENT SUR DISQUE
       JMP  SAUT2
RCOULE BL   @NEWCOL    CHOIX NOUVELLE COULEUR
       JMP  SAUT2
RFOND  MOV  @DP6,@DP6
       JEQ  RFOND1
       CLR  @DP6       ANNULATION REMPLISSAGE FOND
       CLR  @DP1       TRACE CRAYON NON VISIBLE
       LI   R12,MEN110
       MOV  R12,@MEN110+38
       MOV  @SVCOL,@COLORS
       JMP  SAUT2
RFOND1 INC  @DP6             POUR REMPLISSAGE FOND
       INC  @DP1             TRACE CONTINU
       MOV  @COLORS,@SVCOL
       BL   @NEWCOL          CHOIX COULEUR DE FOND
       LI   R12,MENU10
       MOV  R12,@MEN110+38
       JMP  SAUT2
RVISIB MOV  @DP1,@DP1        DRAPEAU TRACE VISIBLE ?
       JNE  RINVIS
       INC  @DP1             DEPLACEMENT VISIBLE DU CRAYON
       MOV  @DP3,@DP3
       JNE  VISIB2
VISIB1 LI   R12,MEN110
       MOV  R12,@MEN110+32
       MOV  R12,@MEN110+34
       JMP  SAUT2
RINVIS CLR  @DP1             SUPRESSION DEPLACEMENT VISIBLE DU CRAYON
       MOV  @DP3,@DP3
       JNE  VISIB1
       JMP  VISIB4
REFFAC MOV  @DP3,@DP3        CHOIX D'EFFACEMENT AVEC CRAYON
       JNE  VISIB3
       INC  @DP3             DRAPEAU=1 :SUPRESSION DE L'EFFACEMENT
       MOV  @DP1,@DP1
```

```
       JEQ  VISIB1
VISIB2 LI   R12,MENU10
       MOV  R12,@MEN110+32
       LI   R12,MEN110
       MOV  R12,@MEN110+34
SAUT2  JMP  ME75
VISIB3 CLR  @DP3             DRAPEAU=0 :EFFACEMENT AU CRAYON
       MOV  @DP1,@DP1
       JNE  VISIB1
VISIB4 LI   R12,MEN110
       MOV  R12,@MEN110+32
       LI   R12,MENU10
       MOV  R12,@MEN110+34
       JMP  ME75
RECRAN BL   @EFFT            EFFACEMENT DE TOUT L'ECRAN
RESE01 JMP  ME75
RON    CLR  @DP6
       JMP  ME75
ROFF   INC  @DP6
       JMP  ME75
RCSURF MOV  @DP5,@DP5
       JEQ  RCSUR1
       LI   R12,MEN130
       MOV  R12,@MEN130+38
       CLR  @DP6
       CLR  @DP5
       CLR  @DP4
       MOV  @SVCOL,@COLORS
       JMP  ME75
RCSUR1 LI   R12,MENU10
       MOV  R12,@MEN130+38
       INC  @DP5
       MOV  @COLORS,@SVCOL
       JMP  ME75
RCSUR2 MOV  @DP5,@DP5
       JEQ  ME75
       INC  @DP4
       INC  @DP3
       CLR  @DP1
       BL   @NEWCOL
       BL   @LIGNE
```

```
       JMP  ME75
RHORIZ MOVB @COORD+1,R1
       MOVB @P1,@COORD+1
       BL   @HORIZ
       MOVB R1,@COORD+1
       JMP  ME75
RVERTI MOVB @COORD,R1
       MOVB @OUTECR,@COORD
       BL   @VERTI
       MOVB R1,@COORD
       JMP  ME75
RSURFA MOV  @COLORS,@SVCOL
       BL   @NEWCOL
       BL   @REMPLI
       MOV  @SVCOL,@COLORS
ME75   NEG  @SVCLAV          INVERSION RTCLAV POUR TEST DANS PROG
ME80   MOV  @SVMENU,@MENU MENU EN COURS
       BL   @AFFMEN          AFFICHAGE MENU EN COURS
       MOV  @RTTM,R11
       LI   R1,>8000         TEMPORISATION
ME85   DEC  R1
       JNE  ME85
       B    *R11
* AFFICHAGE MENU (ADRESSE MENU DANS SAVMEN)
AFFMEN MOV  @MENU,R5  ADRESSE DU MENU
       LI   R4,32     31 CARACTERES A AFFICHER
       LI   R0,>1700  CASE 2 POUR AFFICHAGE
       LI   R2,8      8 BYTES A AFFICHER
ME90   LI   R1,TALPHA ADRESSE DES CARACTERES ASCII
       CLR  R3
       MOVB *R5,R3   CODE ASCII DE LA LETTRE
       SWPB R3       CODE ASCII DANS BYTE FAIBLE
       AI   R3,-32   NUMERO LETTRE DANS TAMPON T
       SLA  R3,3     =*8 (ADRESSE LETTRE DANS T)
       A    R3,R1    ADRESSE LETTRE DANS R1
       BLWP @VMBW    AFFICHAGE DE LA LETTRE
       INC  R5       BYTE SUIVANT (LETTRE SUIVANTE)
       AI   R0,8     ADRESSE CSAE SUIVANTE SUR LIGNE 24
       DEC  R4       LETTRE SUIVANTE
       JNE  ME90     SI NON LETTRE SUIVANTE
       LI   R2,16
```

```
       B    *R11
* EFFACEMENT MASQUE 4 CASES
MASQ   LI   R0,TABLUT+48  LUTIN CARRE
       LI   R1,ATRIBU+48  POSITION PREMIER LUTIN
       LI   R2,16         4 LUTINS (16 BYTES D'ATTRIBUT)
       CLR  R4
       MOVB @OUTECR,R4    POSITION Y HORS ECRAN
       MOVB R4,@ATRIBU+48 LUTIN 1
       MOVB R4,@ATRIBU+52 LUTIN 2
       MOVB R4,@ATRIBU+56 LUTIN 3
       MOVB R4,@ATRIBU+60 LUTIN 4
       BLWP @VMBW         EFFACEMENT DES 4 LUTINS
       B    *R11
```

Source NEWC/S

```
** NEWC/S **
RTNC   DATA 0             POUR RETOUR APPEL
NEWCOL MOV  R11,@RTNC
       MOV  @MASQUE,@MENU POUR LIGNE 24 EN NOIR
       BL   @AFFMEN
* MISE EN PLACE DES COULEURS DANS TABLE/COULEURS
NE05   CLR  R3          NUMERO DU CARRE AFFICHE
       LI   R0,>3708    ADRESSE DEUXIEME CASE
NE10   MOV  @TABCOL(R3),R1  ADRESSE NOUVELLE COULEUR
       CLR  R4          NUMERO DU BYTE (6 BYTES COULEUR,2 BLANC)
NE15   BLWP @VSBW
       INC  R0          BYTE SUIVANT
       INC  R4
       CI   R4,6        DERNIER BYTE COULEUR ?
       JNE  NE15
       AI   R0,8        SAUT D'UNE CASE
       INCT R0          DEBUT DE LA CASE
       INCT R3
       CI   R3,28       DERNIERE COULEUR ?
       JNE  NE10
* AFFICHAGE DES 14 CARRES COULEURS
       LI   R0,>1708    ADRESSE CASE 2
       LI   R2,8        8 BYTES A CHARGER
NE20   LI   R1,LUTINS+88  CARACTERE PLEIN
       BLWP @VMBW
       AI   R0,16       SAUT D'UNE CASE
       DECT R3          NUMERO CASE SUIVANTE
       JNE  NE20        DERNIERE CASE ?
* AFFICHAGE N(NOIR) ET T(TRANSPARENT)
       LI   R0,>17E8       CASE 30
       LI   R1,TALPHA+416 LETTRE T
       LI   R2,8
       BLWP @VMBW
       LI   R0,>17F8       CASE 32
       LI   R1,TALPHA+368 LETTRE N
       BLWP @VMBW
* AFFICHAGE DES 14 REPERES COULEURS
       LI   R0,>1710    ADRESSE CASE 3
       LI   R2,8        8 BYTES A CHARGER
       LI   R3,28
       LI   R1,TALPHA+256  LETTRE @
NE25   BLWP @VMBW
       AI   R0,16       SAUT D'UNE CASE
       AI   R1,8
       DECT R3          NUMERO CASE SUIVANTE
       JNE  NE25        DERNIERE CASE ?
* ATTENTE CRAYON ECLAIRE
       LI   R4,>8000  TEMPO POUR PLACER LE CRAYON
NE30   DEC  R4
       JNE  NE30
NE35   CLR  R3
       MOVB R3,@>8374 SELECTION DE TOUT LE CLAVIER
       BLWP @KSCAN
       MOVB @RTCLAV,R3
       CI   R3,>4000  LETTRE @
       JLT  NE55
       CI   R3,>4E00  LETTRE N
       JLT  NE45
       JGT  NE40
       LI   R3,15     COULEUR NOIRE
       JMP  NE50
NE40   CI   R3,>5400  LETTRE T
       JNE  NE55
       LI   R3,16     COULEUR TRANSPARENT
```

```
       JMP  NE50
NE45   LI   R4,>4000
       S    R4,R3
       SWPB R3
NE50   LI   R4,>0100
       MOVB R4,@>8374
       JMP  NE85
NE55   LI   R4,>0100  SELECTION CLAVIER GAUCHE
       MOVB R4,@>8374
       CLR  R4
       BLWP @KSCAN    TEST SUR CRAYON
       MOVB @RTCLAV,R4  RETOUR CRAYON
       CI   R4,>1200  CRAYON ECLAIRE ?
       JNE  NE35      PAS DE RETOUR CRAYON
* CONTROLE CRAYON SUR LIGNE 24
       MOV  @MASQUE,@MENU
       BL   @AFFMEN
       LI   R4,>2000  TEMPO POUR EXTINCTION CRAYON
NE60   DEC  R4
       JNE  NE60      FIN DE TEMPO ?
       LI   R3,25     TEMPO POUR REPONSE CRAYON
NE65   BLWP @KSCAN    APPEL CRAYON
       MOVB @RTCLAV,R4  RETOUR CRAYON
       CI   R4,>1200  SI CRAYON ENCORE ECLAIRE,MAUVAIS RETOUR CRAYON
       JNE  NE70      RECOMMENCER LA ROUTINE
       B    @NE05
NE70   DEC  R3
       JNE  NE65      FIN DE TEMPO ?
* SEQUENCE RECHERCHE COULEUR CHOISIE
       MOVB @YMENU,@ATRIBU+60  ORDONNEE LUTIN 4 (SUR LIGNE 24)
       LI   R0,>0800  CASE 2 LIGNE 24
       MOVB R0,@ATRIBU+61 ABSCISSE LUTIN 4 (SUR LIGNE 24)
       LI   R0,TABLUT+60  PATRON LUTIN 4 (BLANC SUR CASE TESTEE)
       LI   R1,ATRIBU+60  ATTRIBUTS LUTIN 4
NE75   BLWP @VMBW     AFFICHAGE CARRE
       LI   R5,25     TEMPO REPONSE CRAYON
NE80   BLWP @KSCAN    APPEL CRAYON
       MOVB @RTCLAV,R4  RETOUR CRAYON
       CI   R4,>1200  CRAYON ECLAIRE ?
       JEQ  NE85      COULEUR RETENUE
       DEC  R5        FIN DE TEMPO ?
       JNE  NE80      NOUVEAU TEST CRAYON
       LI   R5,>1000  16 BYTES SUPPLEMENTAIRES (2 CASES A DROITE)
       AB   R5,@ATRIBU+61  2 CASES A DROITE
       INC  R3        NUMERO DE CASE SUIVANT
       CI   R3,16     DERNIERE CASE ?
       JNE  NE75      TEST NOUVELLE CASE
       MOVB @OUTECR,@ATRIBU+60  LUTIN 4 HORS ECRAN
       BLWP @VMBW     EFFACEMENT LUTIN 4
       B    @NE05     PAS TROUVE DE CASE (RETOUR DEBUT ROUTINE)
NE85   MOVB @OUTECR,@ATRIBU+60   LUTIN 4 HORS ECRAN
       BLWP @VMBW     EFFACEMENT LUTIN 4
NE90   INC  R4        REPONSE ACCEPTEE (TEMPO)
       CI   R4,>5000
       JNE  NE90      FIN DE TEMPO ?
       SLA  R3,1      2*R3
       MOV  @TABCOL(R3),@COLORS  NOUVELLE COULEUR
* COULEUR CARACTERES ASCII
       LI   R0,>3700
       LI   R1,>F100
NE95   BLWP @VSBW
       INC  R0
       CI   R0,>3800
       JNE  NE95
       MOV  @RTNC,R11 RETOUR A L'APPEL
       B    *R11
```

Source GRAPH/S

```
RTGRAF DATA 0
CR5    DATA 0,0,0,0
*ROUTINE TRACE DE POINT
POINT  MOV  R11,@RTGRAF  APPEL AVEC Y/X DANS COORD
       LWPI GRAFWS
       MOV  @RTGRAF,R11
       MOVB @COORD+1,R0  X DANS R0
       SRL  R0,8
       MOVB @COORD,R1  Y DANS R1
       SRL  R1,8
       JMP  POINT3
*APPEL DEPUIS ROUTINE GRAPHIQUE CALCULANT UNE COURBE,Y/X DANS SVY/SVX
```

```
POINTC MOV  R11,@RTGRAF
       LWPI GRAFWS
       MOV  @RTGRAF,R11
       MOV  @SVX,R0
       MOV  @SVY,R1
       CI   R0,13      COTE GAUCHE DE L'ECRAN
       JLT  GR20
       CI   R0,252     COTE DROIT DE L'ECRAN
       JGT  GR20
       CI   R1,3       HAUT DE L'ECRAN
       JLT  GR20
       CI   R1,181     BAS DE L'ECRAN
       JGT  GR20
       JMP  POINT3
POINT2 MOV  R11,@RTGRAF  APPEL AVEC Y/X DANS SVY/SVX
       LWPI GRAFWS
       MOV  @RTGRAF,R11
       MOV  @SVX,R0
       MOV  @SVY,R1
POINT3 MOV  R1,R4      APPEL INTERNE A GRAF,Y/X DANS R1/R0
       SLA  R4,5
       SOC  R1,R4
       ANDI R4,>FF07
       MOV  R0,R5
       ANDI R5,7
       A    R0,R4
       S    R5,R4      NUMERO DE L'OCTET DANS R4
       MOV  @DP6,@DP6  REMPLISSAGE FOND ?
       JEQ  GR05       GR05:PAS DE REMPLISSAGE DU FOND
       LI   R5,>4000
       AI   R4,>2000
       SWPB R4         =VSBR
       MOVB R4,@>8C02
       SWPB R4
       MOVB R4,@>8C02
       MOVB @COLORS,R0 EN REMPLACEMENT DE "NOP"
       MOVB @>8800,R6
       ANDI R6,>F000   GARDER COULEUR "ON"
       ANDI R0,>F000   COULEUR "ON"
       SRL  R0,4       TRANSFORMEE EN COULEUR "OFF"
       SOCB R0,R6      COULEURS "ON"+"OFF"
       JMP  GR15
GR05   SWPB R4         =VSBR
       MOVB R4,@>8C02
       SWPB R4
       MOVB R4,@>8C02
       MOV  R5,R0
       MOVB @>8800,R6
       LI   R3,>8000
       SRC  R3,0
       LI   R5,>4000
       MOV  @DP3,@DP3  0:EFFACEMENT;1:PAS D'EFFACEMENT
       JNE  GR10
       SZCB R3,R6      EFFACEMENT:BIT CORRESPONDANT MIS A 0
       JMP  GR15
GR10   SOCB R3,R6      PAS D'EFFACEMENT:BIT CORRESPONDANT MIS A 1
       MOV  R4,R0
       AI   R0,>2000   ADRESSE COULEUR
       SWPB R0         =VSBR
       MOVB R0,@>8C02
       SWPB R0
       MOVB R0,@>8C02
       MOVB @COLORS,R1
       MOVB @>8800,R3
       ANDI R1,>F000   COULEUR "ON"
       ANDI R3,>0F00   COULEUR "OFF"
       SOCB R3,R1      COULEURS "ON"+"OFF"
       SWPB R0         =VSBW
       MOVB R0,@>8C02
       SWPB R0
       SOC  R5,R0
       MOVB R0,@>8C02
       NOP
       MOVB R1,@>8C00
GR15   SWPB R4         =VSBW
       MOVB R4,@>8C02
       SWPB R4
       SOC  R5,R4
       MOVB R4,@>8C02
       NOP
       MOVB R6,@>8C00
GR20   B    *R11
```

```
*ROUTINE TRACE D'AXES
HORIZ  LWPI GRAFWS
       MOVB @COORD,R9  Y
       SRL  R9,8
       LI   R8,13       DEBUT A GAUCHE DE L'ECRAN
GR25   MOV  R8,R0
       MOV  R9,R1
       BL   @POINT3
       INC  R8
       CI   R8,253      COTE DROIT DE L'ECRAN ?
       JLT  GR25
       JMP  GR35
VERTI  LWPI GRAFWS
       MOVB @COORD+1,R8  X
       SRL  R8,8
       LI   R9,3        DEBUT EN HAUT DE L'ECRAN
GR30   MOV  R8,R0
       MOV  R9,R1
       BL   @POINT3
       INC  R9
       CI   R9,182      BAS DE L'ECRAN ?
       JNE  GR30
GR35   LWPI USRWS
       B    *R11
*ROUTINE TRACE DE SEGMENT
LIGNE  MOV  @DP2,@DP2
       JNE  GR40                  1:DEUXIEME POINT DU SEGMENT
       MOV  @COORD,@LIGNES        0:PREMIER POINT DU SEGMENT
       MOV  @P1,@DP2              1 POUR ATTENTE DEUXIEME POINT
       MOVB @YLUTIN+5,@ATRIBU+36 Y POUR AFFICHAGE MIRE FIXE SEGMENT
       MOVB @ATRIBU+13,@ATRIBU+37 X POUR AFFICHAGE MIRE FIXE SEGMENT
       LI   R0,TABLUT+36
       LI   R1,ATRIBU+36
       LI   R2,4
       BLWP @VMBW                 AFFICHAGE MIRE FIXE
       B    *R11
GR40   LWPI GRAFWS
       MOV  @OUTECR,@ATRIBU+36
       LI   R0,TABLUT+36
       LI   R1,ATRIBU+36
       LI   R2,4
       BLWP @VMBW
       MOV  @COORD,@LIGNES+2
       CLR  @DP2
       MOVB @LIGNES+1,R9
       SRL  R9,8
       MOVB @LIGNES+3,R7
       SRL  R7,8
       MOV  R7,@CR5+4
       MOVB @LIGNES,R10
       SRL  R10,8
       INC  R10
       MOVB @LIGNES+2,R8
       SRL  R8,8
       INC  R8
       MOV  R8,@CR5+6
       CLR  R12
       MOV  @P1,@CR5
       MOV  @P1,@CR5+2
       S    R9,R7
       JLT  GR45
       JMP  GR50
GR45   MOV  @M1,@CR5
       NEG  R7
GR50   MOV  R7,R7
       JNE  GR55
       SETO R12
GR55   S    R10,R8
       JLT  GR60
       JMP  GR65
GR60   MOV  @M1,@CR5+2
       NEG  R8
GR65   MOV  R9,R0
       MOV  R10,R1
       DEC  R1
       BL   @POINT3
       C    R9,@CR5+4
       JNE  GR70
       C    R10,@CR5+6
       JEQ  GR80
GR70   MOV  R12,R12
       JLT  GR75
```

```
       A    @CR5,R9
       S    R8,R12
       JMP  GR65
GR75   A    @CR5+2,R10
       A    R7,R12
       JMP  GR65
GR80   LWPI USRWS
       B    *R11
EFFT   LI   R0,>2000   ADRESSE TABLE DES COULEURS
       LI   R1,>1100   NOIR SUR NOIR (LES POINTS RESTENT ALLUMES)
GR85   BLWP @VSBW
       INC  R0
       CI   R0,>3700   23 LIGNES A COLORIER NOIR SUR NOIR
       JNE  GR85
       CLR  R0         DEBUT DE LA TABLE DES PATRONS
       CLR  R1         BYTE FORT=0
GR90   BLWP @VSBW
       INC  R0
       CI   R0,>1700   23 LIGNES A ETEINDRE
       JNE  GR90
       B    *R11
```

Source PROG/S

```
** PROG/S ** SUIVI CRAYON SUR ECRAN **
CRAYON LWPI USRWS     ESPACE DE TRAVAIL PROGRAMME PRINCIPAL
       BL   @MODBIT   PASSAGE EN MODE BIT/MAP
       BL   @CHARLU   CHARGEMENT DES LUTINS DANS LES TABLES VDP
       BL   @TESMEN   AFFICHAGE PREMIER MENU
PR05   CLR  @DP0      DRAPEAU 0:FLASH ETEINT
       BL   @FLASH    FLASH ETEINT
       BL   @MIRE     MIRE AFFICHEE
PR10   LIMI 2         INTERRUPTION ?
       LIMI 0
       INC  @DP0      DRAPEAU 1:FLASH ALLUME
       BL   @FLASH    MIRE ETEINTE,FLASH ALLUME
       LI   R1,100    TEMPORISATION POUR REACTION CRAYON SUR FLASH
PR15   DEC  R1
       JEQ  PR05      POUR NOUVEL APPEL CRAYON OU CLAVIER
       BLWP @KSCAN    APPEL CRAYON OU CLAVIER
       CLR  R0
       MOVB @RTCLAV,R0
       CI   R0,>FF00  TOUCHE ENFONCEE ?
       JEQ  PR15
       MOV  R0,@SVCLAV  SAUVEGARDE R0
       BL   @TESMEN   TEST CRAYON SUR LIGNE 24 ?
       MOV  @SVCLAV,@SVCLAV VALEUR CHANGEE PAR TESMEN ?
       JLT  PR10      SVCLAV<0->UNE OPTION MENU A ETE EXECUTEE
       CLR  @DP0      DRAPEAU POUR ETEINDRE LE FLASH ET POUR RECHERCHER
       BL   @FLASH    LA DIRECTION DE DEPLACEMENT DU CRAYON,
       BL   @MIRE     FLASH ET MIRE ETEINTS
       BL   @CRAY01   RECHERCHE DIRECTION CRAYON
       JMP  PR05      RETOUR NOUVELLE SEQUENCE MIRE/FLASH
MIRE   MOVB @YLUTIN+5,@ATRIBU+40   ORDONNEE POUR MIRE
       MOVB @ATRIBU+13,@ATRIBU+41  ABSCISSE POUR MIRE
       LI   R0,TABLUT+40  ADRESSE MIRE DANS TABLE D'ATTRIBUTS
       LI   R1,ATRIBU+40  NOUVEAUX ATTRIBUTS MIRE
       LI   R2,>0004  4 BYTES A CHARGER
       BLWP @VMBW     NOUVELLE POSITION MIRE DANS TABLE D'ATTRIBUTS
       LI   R3,20000  TEMPORISATION MIRE A L'ECRAN
PR20   DEC  R3
       NOP
       JNE  PR20
       MOV  @OUTECR,@ATRIBU+40  FIN DE TEMPO. -> MIRE HORS ECRAN
       BLWP @VMBW
       B    *R11
FLASH  MOV  @DP0,@DP0 DRAPEAU 0 -> FLASH A ETEINDRE
       JEQ  PR25
       MOVB @YLUTIN+5,@ATRIBU+44    SI NON,ALLUMER FLASH:Y POUR FLASH
       MOVB @ATRIBU+13,@ATRIBU+45                           A L'ECRAN
       JMP  PR30
PR25   MOV  @OUTECR,@ATRIBU+44    FLASH HORS ECRAN
PR30   LI   R0,TABLUT+44  ADRESSE FLASH DANS TABLE D'ATTRIBUTS
       LI   R1,ATRIBU+44  NOUVEAUX ATTRIBUTS FLASH
       LI   R2,>0004  4 BYTES A CHARGER
       BLWP @VMBW
       B    *R11
* TABLE D'ECRAN
MODBIT LI   R0,>01F0  >F0 DANS R1 VDP POUR INHIBER L'ECRAN PENDANT LES
       BLWP @VWTR     OPERATIONS DE MISE EN PLACE DU MODE BIT-MAP
       LI   R0,>1C00  ADRESSE TABLE D'ECRAN
```

```
       CLR  R2
PR35   CLR  R1          DEBUT D'UN DES TROIS TIERS DE LA TABLE D'ECRAN
       CLR  R3
PR40   BLWP @VSBW       NUMERO DE L'OCTET D'ECRAN DANS LA TABLE
       INC  R0
       INC  R3
       AI   R1,>0100    NUMERO SUIVANT POUR OCTET SUIVANT DANS LA TABLE
       CI   R3,>0100    FIN DU TIERS DE L'ECRAN ?
       JNE  PR40
       INC  R2          TIERS SUIVANT
       CI   R2,>0003    TROISIEME TIERS TERMINE ?
       JNE  PR35
* TABLE DES PATRONS
       CLR  R0          ADRESSE DE LA TABLE DES PATRONS
       CLR  R1          TOUS LES BYTES SONT ETEINTS
PR45   BLWP @VSBW
       INC  R0
       CI   R0,>1800    MISE A 0 DE TOUTE LA TABLE DES PATRONS (24 LIGNES)
       JNE  PR45
* TABLE DES COULEURS
       LI   R0,>2000    DEBUT DE LA TABLE DES COULEURS
       LI   R1,>1100    NOIR SUR NOIR
PR50   BLWP @VSBW
       INC  R0
       CI   R0,>3700    23 LIGNES COLORIEES (LIGNE 24 POUR MENU/QUESTIONS)
       JNE  PR50
* COULEUR DES CARACTERES ASCII (LIGNE 24 A L'ECRAN)
       LI   R0,>3700    ADRESSE AFFICHAGE CARACTERES ASCII (LIGNE 24)
       LI   R1,>F100    CARACTERES ASCII NOIR SUR BLANC
PR55   BLWP @VSBW
       INC  R0
       CI   R0,>3800    FIN DE LA LIGNE MENU (LIGNE 24)
       JNE  PR55
*MODIFICATION DES REGISTRES VDP
       LI   R0,>0002    R0=02 <-> BIT6=1 :MODE M3 (BIT-MAP MODE)
       BLWP @VWTR
       LI   R0,>01E0    R1=E0=80+40+20 <-> BITS 0,1,2=1
       BLWP @VWTR       16KRAM,DISPLAY SCREEN,VDP INTERRUPT
       LI   R0,>0207    R2=07 <-> 07*400=1C00
       BLWP @VWTR       ADRESSE TABLE D'ECRAN:>1C00
       LI   R0,>03FF    R3=FF <-> FF*40=...>2000 !!!
       BLWP @VWTR       ADRESSE TABLE DES COULEURS:>2000
       LI   R0,>0403    R4=03 <-> 03*800=...>0000 !!!
       BLWP @VWTR       ADRESSE TABLE DES PATRONS:>0000
       LI   R0,>0537    R5=37 <-> 37*80=1B80
       BLWP @VWTR       ADRESSE TABLE D'ATTRIBUTS DES LUTINS:>1B80
       LI   R0,>0603    R6=03 <-> 03*800=1800
       BLWP @VWTR       ADRESSE TABLE DE DESCRIPTION DES LUTINS:>1800
       LI   R0,>0701    R7=01 <-> ECRAN NOIR
       BLWP @VWTR
       B    *R11
*LUTINS
CHARLU LI   R3,>0100    SELECTION DEMI-CLAVIER GAUCHE
       MOVB R3,@>8374
       LI   R0,>1800    TABLE DE DESCRIPTION DES LUTINS
       LI   R1,LUTINS ADRESSE DES DONNEES DES FORMES DES LUTINS
       LI   R2,152      152 BYTES A CHARGER (19 PATRONS)
       BLWP @VMBW
       LI   R0,>1B80    TABLE DES ATTRIBUTS DES LUTINS
       LI   R1,ATRIBU ADRESSE DES DONNEES DES ATTRIBUTS DES LUTINS
       LI   R2,128      32 LUTINS
       BLWP @VMBW
       B    *R11
       END  CRAYON
```

Code machine à entrer en mémoire avec EASY BUG (option M)

```
7000   A5 5A 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7010   00 00 00 00 00 00 00 00 00 00 00 00 71 18 7F F8
7020   7E 22 A0 00 FF E0 20 00 3F FF 00 36 10 01 06 1C
7030   11 00 50 5C 00 04 10 0E 00 01 3A 20 20 20 20 20
7040   20 20 39 39 2F 34 20 41 53 20 20 20 20 20 20 20
7050   20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
7060   20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
7070   20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
7080   20 20 20 20 20 20 30 30 38 31 44 53 4B 31 2E 00
7090   00 00 00 00 72 48 00 00 10 01 00 00 70 8F 00 0F
```

```
70A0   00 05 00 08 00 00 00 00 61 36 10 01 70 B8 7E B8
70B0   90 00 70 D8 62 FE C0 00 1B A8 72 44 00 04 00 00
70C0   00 00 72 E5 00 06 F8 00 00 00 00 01 00 00 7E 3E
70D0   00 00 00 00 00 00 00 00 2D 0F 7F F8 00 4F 00 3A
70E0   7F F8 A0 00 6F 06 7F B8 00 3A 00 00 64 76 63 5A
70F0   00 01 70 B8 7E 22 C0 00 00 00 00 00 00 00 00 00
7100   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7110   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7120   FF FF C7 B7 B7 B7 CB FF FF FF C7 BB BF BB C7 FF
7130   F3 F7 C7 B7 B7 B7 C3 FF FF FF C7 BB 83 BF C7 FF
7140   E7 DB DF 8F DF DF DF FF 9F DF D7 CB DB DB DB FF
7150   EF FF 8F EF EF EF 83 FF FF FF 57 AB AB AB AB FF
7160   FF FF 87 DB DB C7 DF 8F FF FF A7 DB DF DF DF FF
7170   FF FF C7 BF C7 FB 87 FF DF DF 87 DF DF DB E7 FF
7180   FF FF BB BB D7 D7 EF FF FF E7 D7 F7 F7 F7 E3 FF
7190   FF E7 DB FB F7 EF C3 FF FF E3 FB F7 FB DB E7 FF
71A0   FF F7 E7 D7 D7 C3 F7 FF FF E3 EF E7 FB DB E7 FF
71B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
71C0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
71D0   00 00 00 00 00 00 00 00 20 00 00 10 20 30 40 50
71E0   60 70 80 90 A0 B0 C0 D0 E0 F0 72 DA 71 EE 20 20
71F0   20 20 20 20 20 20 20 20 20 08 38 68 98 C8 F8 B7
7200   AE AA A5 A5 A5 AA AE AE AE C0 C0 00 22 20 21 01
7210   11 10 12 02 00 00 00 AE 1B 00 00 C0 1C 00 0F
7220   C0 1C 01 0F C0 1C 02 0F C0 18 03 0F C0 13 04 0F
7230   C0 13 05 0F C0 13 06 0F C0 18 07 0F C0 18 0A 01
7240   C0 00 08 0F C0 00 08 08 C0 00 09 0F C0 F8 0B 0F
7250   C0 F8 0B 0F C0 F8 0B 0F C0 F8 0B 0F D0 00 20 30
7260   7C 7F 7F 3F 3F 1F 03 0F 3F 3F 1F 0F 03 00 1F 3F
7270   3F 7F 7F FC 30 00 FE FC 7C 78 38 30 00 00 F8 FC
7280   FC FE FE 3E 0C 04 C0 F0 F8 FC FC F0 C0 00 00 0C
7290   3F FE FE FC FC F8 00 00 18 38 3C 7C 7E FE 10 10
72A0   00 C6 00 10 10 00 00 00 38 38 38 00 00 00 EE FE
72B0   FE 7C FE FE EE 00 FF FF FF FF FF FF FF FF 00 00
72C0   00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00
72D0   00 00 00 01 FF FF 72 DA 7E 32 29 2E 2C 2F 23 30
72E0   22 31 24 32 2C 20 72 DA 72 FE 73 22 73 46 72 DA
72F0   72 DA 74 92 74 FE 74 FE 74 FE 75 06 74 FE 2D 2E
7300   27 2F 22 30 25 31 2A 32 2C 20 72 FE 72 FE 72 DA
7310   72 FE 72 FE 72 DA 74 DE 74 F2 74 AC 74 B2 75 86
7320   74 FE 21 2E 2B 2F 26 30 2D 31 20 20 2C 20 73 22
7330   73 22 73 22 73 22 73 22 72 DA 74 9C 74 A6 75 5E
7340   75 72 75 06 74 FE 21 2E 24 2F 20 20 2B 31 29 32
7350   2C 20 73 46 73 46 73 46 73 46 73 46 72 DA 75 0C
7360   75 12 75 0A 75 18 75 40 74 FE C8 0B 72 D8 C8 20
7370   71 EA 72 D6 C8 20 71 EC 71 EA 06 A0 75 B6 C8 20
7380   72 D6 71 EA D1 20 71 FF D8 04 72 4C D8 04 72 50
7390   D8 04 72 54 D8 04 72 58 02 00 1B B0 02 01 72 4C
73A0   02 04 03 E8 06 04 16 FE 02 04 00 14 04 C5 06 04
73B0   13 09 04 20 60 20 D1 60 83 75 02 85 12 00 16 F7
73C0   04 60 75 9E 04 C6 04 C7 D1 E6 71 F9 D8 07 72 4D
73D0   02 27 08 00 D8 07 72 51 02 27 08 00 D8 07 72 55
73E0   02 27 08 00 D8 07 72 59 04 20 60 28 02 04 00 18
73F0   04 20 60 20 D1 60 83 75 02 85 12 00 13 3C 02 85
7400   13 00 16 03 02 06 00 00 10 36 02 85 07 00 16 03
7410   02 06 00 01 10 30 02 85 08 00 16 03 02 06 00 02
7420   10 2A 02 85 09 00 16 03 02 06 00 03 10 24 02 85
7430   0A 00 16 03 02 06 00 04 10 1E 02 85 0B 00 16 03
7440   02 06 00 05 10 18 06 04 16 D3 05 86 02 86 00 05
7450   11 BB 13 04 06 A0 76 00 04 60 75 9E 02 07 F8 00
7460   D8 07 72 4D D8 07 72 51 D8 07 72 55 D8 07 72 59
7470   04 20 60 28 10 BB 06 A0 76 00 C1 20 72 D6 02 24
7480   00 0C 0A 16 A1 06 C8 14 72 D6 02 24 00 0C C0 54
7490   04 51 06 A0 7B C4 02 E0 70 B8 10 31 06 A0 7C DE
74A0   06 A0 7C FE 10 2C 06 A0 7D 22 10 29 06 A0 7B 6A
74B0   10 26 C8 20 72 CA 72 CA 13 08 04 E0 72 CA 04 E0
74C0   72 C0 C8 20 71 D6 71 D8 10 1A 05 A0 72 CA 05 A0
74D0   72 C0 C8 20 71 D8 71 D6 06 A0 7B 6A 10 10 C8 20
74E0   72 C0 72 C0 16 03 05 A0 72 C0 10 09 04 E0 72 C0
74F0   10 06 C8 20 72 C4 72 C4 16 03 05 A0 72 C4 10 4D
7500   04 E0 72 C4 10 4A 06 A0 7D FC 10 47 04 E0 72 CA
7510   10 44 05 A0 72 CA 10 41 C8 20 72 C8 72 C8 13 0A
7520   04 E0 72 CA 04 E0 72 C8 04 E0 72 C6 C8 20 71 D6
7530   71 D8 10 33 05 A0 72 C8 C8 20 71 D8 71 D6 10 2D
7540   C8 20 72 C8 72 C8 13 29 05 A0 72 C6 05 A0 72 C4
7550   04 E0 72 C0 06 A0 7B 6A 06 A0 7D 22 10 1E D0 60
7560   72 19 D8 20 72 D2 72 19 06 A0 7C DE D8 01 72 19
7570   10 14 D0 60 72 18 D8 20 72 0A 72 18 06 A0 7C FE
7580   D8 01 72 18 10 0A C8 20 71 D8 71 D6 06 A0 7B 6A
7590   06 A0 78 E8 C8 20 71 D6 71 D8 05 20 71 D0 C8 20
75A0   72 D6 71 EA 06 A0 75 B6 C2 E0 72 D8 02 01 80 00
75B0   06 01 16 FE 04 5B C1 60 71 EA 02 09 00 02 02 04
```

```
75C0   00 0B 02 00 17 08 02 02 00 08 02 01 71 18 04 C3
75D0   D0 D5 06 C3 02 23 FF E0 0A 33 A0 43 04 20 60 28
75E0   05 85 06 09 13 07 02 20 00 08 06 04 16 EE 02 02
75F0   00 10 04 5B 02 09 00 02 02 20 00 28 06 04 10 E5
7600   02 00 1B B0 02 01 72 4C 02 02 00 10 04 C4 D1 20
7610   72 0A D8 04 72 4C D8 04 72 50 D8 04 72 54 D8 04
7620   72 58 04 20 60 28 04 5B 00 00 00 00 C8 0B 76 28
7630   C1 20 71 D0 02 84 12 00 13 60 04 C9 D8 20 72 05
7640   72 44 D8 20 72 29 72 45 02 00 1B A8 02 01 72 44
7650   02 02 00 04 04 20 60 28 02 08 07 D0 02 07 02 58
7660   04 20 60 20 D8 20 83 7C 83 7C 16 15 D1 20 83 75
7670   02 84 FF 00 13 0C 02 89 00 03 11 05 02 07 00 C8
7680   06 07 16 FE 10 09 06 07 16 EB 05 89 10 05 06 08
7690   16 E7 04 60 77 BA 04 C9 D1 20 83 75 16 03 02 03
76A0   00 07 10 28 02 84 0E 00 16 02 04 C3 10 23 02 84
76B0   03 00 16 03 02 03 00 01 10 1D 02 84 06 00 16 03
76C0   02 03 00 02 10 17 02 84 05 00 16 03 02 03 00 03
76D0   10 11 02 84 04 00 16 03 02 03 00 04 10 0B 02 84
76E0   02 00 16 03 02 03 00 05 10 05 02 84 0F 00 16 04
76F0   02 03 00 06 06 A0 77 C0 10 A1 02 02 00 04 02 0A
7700   00 3F 04 CC 02 0D 00 04 D8 20 72 01 72 3C 02 00
7710   1B 80 02 01 72 1C 04 C3 C1 C0 C2 01 02 09 00 14
7720   02 00 1B A0 02 01 72 3C 04 20 60 28 C0 07 C0 48
7730   D4 63 72 00 04 20 60 28 04 C4 04 20 60 20 D1 20
7740   83 75 02 84 12 00 13 23 06 09 16 F7 C0 07 C0 48
7750   D4 60 72 0A 04 20 60 28 06 0A 13 22 C3 0C 13 10
7760   04 CC C0 C3 13 04 06 03 61 CD 62 0D 10 D7 02 00
7770   1B 9C 02 01 72 38 02 03 00 07 C1 C0 C2 01 10 CE
7780   02 83 00 07 13 C4 05 83 A1 CD A2 0D 10 C7 06 A0
7790   77 C0 02 0A 00 27 05 8C 02 09 00 32 04 60 77 20
77A0   02 00 80 00 06 00 16 FE D8 20 72 0A 72 3C 02 00
77B0   1B A0 02 01 72 3C 04 20 60 28 C2 E0 76 28 04 5B
77C0   C8 0B 76 2A 04 C0 D0 23 72 0C C1 00 02 44 02 00
77D0   16 0A 02 06 72 00 02 01 FF FF 02 04 03 00 02 05
77E0   00 01 06 A0 78 80 C1 00 02 44 01 00 16 0A 02 06
77F0   72 00 02 01 00 01 02 04 B5 00 02 05 00 01 06 A0
7800   78 80 C1 00 02 44 20 00 16 0A 02 06 72 19 02 01
7810   FF FF 02 04 0D 00 02 05 00 04 06 A0 78 80 C1 00
7820   02 44 10 00 16 0A 02 06 72 19 02 01 00 01 02 04
7830   FC 00 02 05 00 04 06 A0 78 80 D8 20 72 08 72 18
7840   D8 20 72 01 72 3C D8 20 72 29 72 3D C8 00 71 D4
7850   C8 01 71 D2 C8 20 72 C0 72 C0 13 05 06 A0 7B C4
7860   02 E0 70 B8 10 0A C8 20 72 C6 72 C6 13 06 05 A0
7870   72 C2 06 A0 7D 22 04 E0 72 C2 C2 E0 76 2A 04 5B
7880   91 16 13 07 02 04 00 09 06 C1 B5 81 A1 85 06 04
7890   16 FC 04 5B 00 00 00 00 00 00 00 00 00 00 00 00
78A0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
78B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
78C0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
78D0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
78E0   00 00 00 00 00 00 00 C8 0B 78 94 02 E0 71 B0
78F0   C8 20 72 18 78 A8 B8 20 72 D3 78 A8 C8 20 72 D2
7900   78 AA 02 03 00 04 02 23 FF FC 11 78 04 C0 D0 23
7910   78 A9 06 C0 04 C1 D0 63 78 A8 06 C1 C8 23 78 AA
7920   78 9E C8 00 78 A4 C8 00 78 A2 C2 80 06 A0 7A 68
7930   13 65 06 A0 7A 06 02 81 00 04 11 E5 02 81 00 B4
7940   15 E2 C8 20 72 D4 78 A0 88 20 78 A4 78 A6 13 1B
7950   88 20 78 A4 78 A6 15 08 C2 A0 78 A6 C3 20 78 A4
7960   06 0C 06 A0 7A F8 10 0F C2 A0 78 A4 C3 20 78 A6
7970   05 20 78 9E A0 60 78 9E 06 0C 06 A0 7A F8 05 20
7980   78 9E A0 60 78 9E C8 20 78 A6 78 A4 A0 60 78 9E
7990   C2 A0 78 A2 06 A0 7A 68 13 1F C8 20 72 D2 78 A0
79A0   02 0C 00 FD 05 20 78 9E A2 A0 78 A0 83 0A 13 03
79B0   06 A0 7A 68 16 F9 C3 20 78 A2 06 0A C8 0A 78 A2
79C0   05 8A A0 60 78 9E 05 20 78 A0 06 A0 7A F8 05 20
79D0   78 9E A0 60 78 9E 10 AD C8 20 72 D4 78 A0 02 0C
79E0   00 0D A2 A0 78 A0 83 0A 13 03 06 A0 7A 68 13 F9
79F0   88 0A 78 A4 11 88 C8 0A 78 A2 10 9B 02 E0 70 B8
7A00   C2 E0 78 94 04 5B C8 0B 78 98 C0 20 78 A2 C0 80
7A10   C2 80 06 A0 7A 68 06 A0 7A 9A 02 84 01 00 16 1E
7A20   02 27 FF F8 06 A0 7B 36 02 82 00 10 11 13 C2 49
7A30   16 11 02 0A FF 00 06 A0 7B 4C D2 A0 71 D8 02 27
7A40   20 00 06 A0 7B 4C 02 27 E0 00 02 27 FF F8 02 22
7A50   FF F8 10 E8 02 04 00 01 06 A0 7A 9A 05 82 C8 02
7A60   78 A6 C2 E0 78 98 04 5B C8 0B 78 9C C0 0A C1 C1
7A70   0A 57 E1 C1 02 47 FF 07 C1 00 02 44 00 07 A1 C0
7A80   61 C4 06 A0 7B 36 02 04 00 80 02 40 00 07 13 01
7A90   08 04 C2 E0 78 9C 41 09 04 5B 41 09 13 0C E2 44
7AA0   06 02 02 82 00 0C 15 02 05 82 10 05 0A 14 02 84
7AB0   01 00 13 01 10 F2 C8 0B 78 96 C2 89 06 CA 06 A0
7AC0   7B 4C 10 02 C8 0B 78 96 02 27 20 00 06 C7 D8 07
7AD0   8C 02 06 C7 D8 07 8C 02 D2 A0 71 D8 D3 60 88 00
```

```
7AE0   02 4A F0 00 02 4D 0F 00 F2 8D 06 A0 7B 4C 02 27
7AF0   E0 00 C2 E0 78 96 04 5B C8 0B 78 9A A2 A0 78 A0
7B00   83 0A 13 16 06 A0 7A 68 13 F9 C8 CA 78 A8 06 C1
7B10   D8 C1 78 A8 06 C1 C8 E0 78 9E 78 AA 02 23 00 04
7B20   A2 A0 78 A0 83 0A 13 04 06 A0 7A 68 16 F9 10 E6
7B30   C2 E0 78 9A 04 5B 06 C7 D8 07 8C 02 06 C7 D8 07
7B40   8C 02 04 C9 D2 60 88 00 06 C9 04 5B C2 07 02 0C
7B50   40 00 06 C8 D8 08 8C 02 06 C8 E2 0C D8 08 8C 02
7B60   10 00 D8 0A 8C 00 04 5B 00 00 C8 0B 7B 68 04 C3
7B70   D8 03 83 74 04 20 60 20 D0 E0 83 7C 13 F8 D0 E0
7B80   83 75 0B 83 02 83 00 30 11 F2 02 83 00 39 12 09
7B90   02 83 00 41 11 EC 02 83 00 46 15 E9 02 23 FF C9
7BA0   10 02 02 23 FF D0 D8 23 71 DA 71 D8 02 03 01 00
7BB0   D8 03 83 74 C2 E0 7B 68 04 5B 00 00 00 00 00 00
7BC0   00 00 00 00 C8 0B 7B BA 02 E0 71 B0 C2 E0 7B BA
7BD0   D0 20 72 19 09 80 D0 60 72 18 09 81 10 21 C8 0B
7BE0   7B BA 02 E0 71 B0 C2 E0 7B BA C0 20 71 D4 C0 60
7BF0   71 D2 02 80 00 0D 11 72 02 80 00 FC 15 6F 02 81
7C00   00 03 11 6C 02 81 00 B5 15 69 10 0A C8 0B 7B BA
7C10   02 E0 71 B0 C2 E0 7B BA C0 20 71 D4 C0 60 71 D2
7C20   C1 01 0A 54 E1 01 02 44 FF 07 C1 40 02 45 00 07
7C30   A1 00 61 05 C8 20 72 CA 72 CA 13 15 02 05 40 00
7C40   02 24 20 00 06 C4 D8 04 8C 02 06 C4 D8 04 8C 02
7C50   D0 20 71 D8 D1 A0 88 00 02 46 F0 00 02 40 F0 00
7C60   09 40 F1 80 10 31 06 C4 D8 04 8C 02 06 C4 D8 04
7C70   8C 02 C0 05 D1 A0 88 00 02 03 80 00 0B 03 02 05
7C80   40 00 C8 20 72 C4 72 C4 16 02 51 83 10 1D F1 83
7C90   C0 04 02 20 20 00 06 C0 D8 00 8C 02 06 C0 D8 00
7CA0   8C 02 D0 60 71 D8 D0 E0 88 00 02 41 F0 00 02 43
7CB0   0F 00 F0 43 06 C0 D8 00 8C 02 06 C0 E0 05 D8 00
7CC0   8C 02 10 00 D8 01 8C 00 06 C4 D8 04 8C 02 06 C4
7CD0   E1 05 D8 04 8C 02 10 00 D8 06 8C 00 04 5B 02 E0
7CE0   71 B0 D2 60 72 18 09 89 02 08 00 0D C0 08 C0 49
7CF0   06 A0 7C 20 05 88 02 88 00 FD 11 F8 10 0F 02 E0
7D00   71 B0 D2 20 72 19 09 88 02 09 00 03 C0 08 C0 49
7D10   06 A0 7C 20 05 89 02 89 00 B6 16 F8 02 E0 70 B8
7D20   04 5B C8 20 72 C2 72 C2 16 15 C8 20 72 18 72 14
7D30   C8 20 72 D2 72 C2 D8 20 72 05 72 40 D8 20 72 29
7D40   72 41 02 00 1B A4 02 01 72 40 02 02 00 04 04 20
7D50   60 28 04 5B 02 E0 71 B0 C8 20 72 0A 72 40 02 00
7D60   1B A4 02 01 72 40 02 02 00 04 04 20 60 28 C8 20

7D70   72 18 72 16 04 E0 72 C2 D2 60 72 15 09 89 D1 E0
7D80   72 17 09 87 C8 07 7B C0 D2 A0 72 14 09 8A 05 8A
7D90   D2 20 72 16 09 88 05 88 C8 08 7B C2 04 CC C8 20
7DA0   72 D2 7B BC C8 20 72 D2 7B BE 61 C9 11 01 10 04
7DB0   C8 20 72 D4 7B BC 05 07 C1 C7 16 01 07 0C 62 0A
7DC0   11 01 10 04 C8 20 72 D4 7B BE 05 08 C0 09 C0 4A
7DD0   06 01 06 A0 7C 20 88 09 7B C0 16 03 88 0A 7B C2
7DE0   13 0A C3 0C 11 04 A2 60 7B BC 63 08 10 EF A2 A0
7DF0   7B BE A3 07 10 EB 02 E0 70 B8 04 5B 02 00 20 00
7E00   02 01 11 00 04 20 60 24 05 80 02 80 37 00 16 FA
7E10   04 C0 04 C1 04 20 60 24 05 80 02 80 17 00 16 FA
7E20   04 5B 02 E0 70 B8 06 A0 7E E8 06 A0 7F 8E 06 A0
7E30   73 6A 04 E0 72 BE 06 A0 7E BA 06 A0 7E 88 03 00
7E40   00 02 03 00 00 00 05 A0 72 BE 06 A0 7E BA 02 01
7E50   00 64 06 01 13 EE 04 20 60 20 04 C0 D0 20 83 75
7E60   02 80 FF 00 13 F6 C8 00 71 D0 06 A0 73 6A C8 20
7E70   71 D0 71 D0 11 E4 04 E0 72 BE 06 A0 7E BA 06 A0
7E80   7E 88 06 A0 76 2C 10 D5 D8 20 72 05 72 44 D8 20
7E90   72 29 72 45 02 00 1B A8 02 01 72 44 02 02 00 04
7EA0   04 20 60 28 02 03 4E 20 06 03 10 00 16 FD C8 20
7EB0   72 0A 72 44 04 20 60 28 04 5B C8 20 72 BE 72 BE
7EC0   13 07 D8 20 72 05 72 48 D8 20 72 29 72 49 10 03
7ED0   C8 20 72 0A 72 48 02 00 1B AC 02 01 72 48 02 02
7EE0   00 04 04 20 60 28 04 5B 02 00 01 F0 04 20 60 34
7EF0   02 00 1C 00 04 C2 04 C1 04 C3 04 20 60 24 05 80
7F00   05 83 02 21 01 00 02 83 01 00 16 F7 05 82 02 82
7F10   00 03 16 F1 04 C0 04 C1 04 20 60 24 05 80 02 80
7F20   18 00 16 FA 02 00 20 00 02 01 11 00 04 20 60 24
7F30   05 80 02 80 37 00 16 FA 02 00 37 00 02 01 F1 00
7F40   04 20 60 24 05 80 02 80 38 00 16 FA 02 00 00 02
7F50   04 20 60 34 02 00 01 E0 04 20 60 34 02 00 02 07
7F60   04 20 60 34 02 00 03 FF 04 20 60 34 02 00 04 03
7F70   04 20 60 34 02 00 05 37 04 20 60 34 02 00 06 03
7F80   04 20 60 34 02 00 07 01 04 20 60 34 04 5B 02 03
7F90   01 00 D8 03 83 74 02 00 18 00 02 01 72 5E 02 02
7FA0   00 60 04 20 60 28 02 00 1B 80 02 01 72 1C 02 02
7FB0   00 40 04 20 60 28 04 5B 00 00 00 00 00 00 00 00
7FC0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7FD0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7FE0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7FF0   00 00 00 00 00 00 00 00 43 52 41 59 4F 4E 7E 22
```

Le traitement des fichiers en PASCAL

Gérard Santraille

Les adversaires du Pascal argueraient, à juste titre d'ailleurs, que ce langage pêche par sa lourdeur au niveau des manipulations de fichiers. La première version de Pascal (proposée par N. Wirth dans les années 1970) était pratiquement dépourvue d'instructions permettant de gérer efficacement les entrées/sorties sur fichiers.
Depuis ces temps héroïques, Pascal s'est considérablement étoffé et tous les compilateurs offrent désormais de nombreuses commandes permettant l'accès à de très nombreux types de fichiers : nous verrons par la suite ceux qui sont prédéfinis. Quant aux méthodes d'accès, elles se répartissent très classiquement en deux catégories :

- **accès séquentiel** : les données sont lues et/ou écrites les unes après les autres.
- **accès direct** : on se positionne directement sur un enregistrement pour lire ou écrire en spécifiant sa position par rapport au début du fichier.

Toutes les autres techniques, plus ou moins élaborées, d'accès aux fichiers comme le *séquentiel indexé* ou le *Hash-coding* s'appuient sur ces deux types fondamentaux.

Notions de bases :

Fichiers
Enregistrements
Tampons

Un fichier est un ensemble (de données) structuré en enregistrements. On symbolise souvent un fichier par une feuille de papier et ses enregistrements par les lignes formant le texte (contenu du fichier).
Prenons un exemple concret. Supposons que l'on désire collecter des informations sur un grand nombre de personnes (fichier de relations). On dispose pour chaque individu de son nom, de son prénom et de son adresse. Manuellement on écrirait sur une feuille, pour chaque personne, la liste des informations connues.
La déclaration Pascal correspondant à une personne pourrait être :

```
type
personne = packed record of
        nom      : string[20];
        prenom   : string[15];
        tel_dom  : string[13];
        tel_bur  : string[22];
        adr1     : string[25];
        adr2     : string[20];
        adr3     : string[10];
        code     : string[6];
        ville    : string[20];
        end;
```

Cet ensemble d'informations, relatif à une seule personne, correspond à un enregistrement (**record**).

A l'attention des puristes signalons que le terme **Enregistrement** est ici, propre au Pascal. Il ne correspond pas à la notion "d'enregistrement de fichier" communément admise par les informaticiens.

Dans notre cas l'enregistrement est une entité dont l'organisation (le **type**) est entièrement définissable par l'utilisateur. Ce dernier est donc libre de créer la structure qui correspondra le mieux à son problème. La taille et la complexité d'un 'record' n'ont pour limite que l'imagination du programmeur.

L'assemblage de tous les enregistrements va former un fichier. Un fichier est donc une suite d'enregistrements qui :
- ont tous la même structure (le même type)
- peuvent eux-même être composés de plusieurs éléments (appelés **champs**).

La déclaration d'un tel fichier se fait dans la rubrique **Var** par :
 fichier : **file** of **personne**;
Notons que la définition du type fichier ne précise que le type des éléments qui le compose.

Remarque : Un enregistrement peut lui-même être composé d'enregistrements. On peut ainsi bâtir des structures imbriquées extrêmement complexes. La composition des données en ensembles puis en ensembles d'ensembles est l'une des grandes originalités du Pascal.

La déclaration d'une variable fichier **F** introduit automatiquement une variable associée au fichier. Cette variable s'appelle le **tampon** (ou la **fenêtre**) de fichier et son type est celui des composants du fichier. Littéralement le fichier est désigné par **F^**.

Si nous reprenons notre feuille de papier pour représenter un fichier, la fenêtre correspondrait à une règle que l'on déplacerait verticalement de ligne en ligne. Dans cette représentation chaque ligne équivaut à un enregistrement. A un instant donné, on ne peut accéder pour lire ou écrire qu'à l'enregistrement situé dans la fenêtre (la ligne pointée par la règle). Cet enregistrement est appelé **élément courant**. Une opération de lecture (ou d'écriture) sur le fichier à pour action de déplacer l'élément courant dans le tampon du fichier.

Toutes les actions d'entrées/sorties se font par l'intermédiaire de ce tampon. Ce dernier se comporte comme une variable du type des enregistrements du fichier. Les accès à un fichier (sauf comme nous le verrons plus loin pour certains fichiers de type prédéfinis) se décomposent de la façon suivante :

Lecture	Ecriture
• positionnement du tampon à l'endroit souhaité	• positionnement du tampon à l'endroit souhaité
pour l'accès direct seulement	
• lecture du fichier vers la variable tampon	• écriture (assignation) de la variable tampon
• visualisation de la variable tampon	• écriture physique sur le fichier

Instructions élémentaires

Lecture

Reprenons notre fichier **F** : file of personne avec :

```
personne = packed record of
    nom       : string[20];
    prenom    : string[15];
    tel_dom   : string[13];
    tel_bur   : string[22];
    adr1      : string[25];
    adr2      : string[20];
    adr3      : string[10];
    code      : string[6];
    ville     : string[20];
    end;
```

L'ouverture d'un fichier en lecture se fait au moyen de l'instruction **RESET(F,'nom_du_fichier')** ou **RESET(F)** qui est équivalent à **RESET(F,'F')**. Cette opération positionne la fenêtre sur le premier élément du fichier et le tampon **F^** est initialisé avec la valeur de l'élément courant placé sous la fenêtre.
Lors de l'ouverture par RESET le premier élément est automatiquement lu. La variable tampon contient donc le premier enregistrement : il n'est pas nécessaire de le lire.
On peut manipuler les différents éléments de cet enregistrement de la même façon qu'une variable classique du même type. Ainsi si l'on désire visualiser à l'écran le champ de l'enregistrement contenant le nom de la personne il suffit d'écrire **WRITELN(F^.NOM)**.

Pour accéder à l'enregistrement suivant du fichier il faut exécuter la commande **GET(F)**. Cette instruction a deux effets :

- la fenêtre de lecture avance d'un enregistrement
- la variable F^ prend les nouvelles valeurs de l'enregistrement pointé par la fenêtre.

la lecture d'un enregistrement se fait donc en deux étapes.

- transfert des informations de l'enregistrement vers la variable tampon F^
- visualisation de cette variable à l'écran.

On peut de cette façon parcourir *séquentiellement* tout le fichier et l'afficher à l'écran. Lorsque la fin du fichier est atteinte la fonction intrinsèque **EOF(F)** devient vraie.
La procédure suivante permet de lire en entier notre fichier et d'afficher les noms de chaque enregistrement.

```
procedure LireFichier (var fichier : file
of personne);

begin
reset(f,fichier);
while not eof(f) do
  begin
  writeln(f^.nom);
  get(f)
  end;
close(f,lock)
end.
```

Notons la présence du **while not eof(f)** qui permet d'effectuer une séquence d'instructions tant que eof(f) est *faux* donc tant que la fin du fichier n'a pas été atteinte.

Remarque : L'ouverture d'un fichier au moyen de RESET permet également de le mettre à jour et donc de modifier certains enregistrements ou d'en créer de nouveaux. Par contre RESET ne permet pas de créer un fichier : il faut que le fichier soit déjà physiquement présent sur la disquette pour pouvoir être ouvert par RESET.

Ecriture

Comme pour la lecture, la première opération doit être l'ouverture du fichier. Cette action associe, comme nous l'avons vu, un tampon au fichier qui sera utilisé pour les opérations d'entrées/sorties.
L'écriture sur un fichier se fera en deux étapes.

- écriture dans la variable tampon F^ (affectation)
- écriture de F^ sur le fichier.

L'instruction qui permet d'ouvrir un fichier en écriture est :
REWRITE(F,'nom_du_fichier') ou
REWRITE(F) qui est équivalent à
REWRITE(F,'F').
Celle qui écrit la variable tampon F^ sur le fichier est **PUT(F)**. Par ailleurs, REWRITE est le mode d'ouverture qui doit être utilisé pour créer un nouveau fichier.
Ainsi si l'on désire écrire sur le fichier l'enregistrement correspondant à une personne il convient de définir chacun des éléments de F^ puis d'écrire la variable.
Exemple :

```
  ...............
  f^.nom        := 'Jackson';
  f^.prenom     := 'Michael';
  f^.tel_dom    := '(43) 66.77.88';
  f^.tel_bur    := '(8) 22.11.22';
  f^.adr1       := '28, Melody Street';
  f^.adr2       := 'flat 23WZ, 1024th
floor';
  f^.adr3       := '';
  f^.code       := '43.122';
  f^.ville      := 'THRILLER CITY';

  put(f);
  ...............
```

Les deux instructions PUT/GET constituent les deux éléments de base pour pouvoir dialoguer avec un fichier. Nous venons de voir comment ouvrir, lire et écrire dans un fichier. Une fois le travail terminé il convient de le fermer. La fermeture d'un fichier provoque plusieurs actions :

- dissociation de la variable F associée à 'nom_de_fichier' lors du Reset(F,'nom_du_fichier').
- restitution au système de la mémoire utilisée par la variable tampon F^.

C'est l'instruction **CLOSE(F,option)** qui a cette lourde tâche. Le mot-clé option permet d'effectuer divers types de fermeture.

LOCK : c'est l'option que l'on emploie dans la majorité des cas. Le fichier est fermé et toutes les opérations sauvegardées.

PURGE : à manipuler avec beaucoup de précautions. Le fichier est détruit qu'il ait été ouvert en lecture ou en écriture.

CRUNCH : le fichier est fermé et tronqué à l'endroit où se trouvait la fenêtre. Toutes les informations situées au delà sont perdues.

NORMAL : si le fichier a été ouvert en lecture il est fermé et aucune action particulière n'a lieu. S'il a été ouvert en écriture il est détruit comme avec PURGE.

Si le mot-clé option est absent l'option par défaut est NORMAL.

Différents modes d'accès

Toutes les opérations que nous venons de décrire utilisent l'accès séquentiel (le plus simple mais bien souvent le moins performant). Il existe de nombreuses techniques permettant d'isoler une information noyée dans une masse importante.

Si l'on cherche la signification du mot **Seringa** dans le dictionnaire (plante à fleurs blanches très odorantes) il est bien évident que l'on ne va pas parcourir l'ouvrage de la première page à celle contenant notre cible. On effectue une recherche par encadrements successifs du mot à trouver. Cette technique qui s'apparente à la dichotomie est évidemment beaucoup plus efficace.

Une première remarque s'impose. Une recherche autre que séquentielle ne peut se faire que sous certaines conditions. Par exemple on pressent bien qu'une recherche par encadrements successifs ne peut se faire que si l'ensemble de travail est ordonné.

Plus généralement, toutes les méthodes d'accès à des informations contenues dans un fichier imposent certaines restrictions et ne peuvent s'appliquer à tous les types d'ensembles.

Pour pouvoir gagner du temps lors de l'accès il faut inévitablement faire l'une (ou les deux) de ces concessions :

- Fichier à structure particulière (comme le dictionnaire). Le domaine sur lequel va s'effectuer la recherche doit présenter certaines caractéristiques particulières. Il sera partiellement ou totalement ordonné ou encore ses éléments seront regroupés par famille suivant un critère de sélection.
- Stockage d'informations supplémentaires.

Reprenons l'exemple de le recherche d'un mot dans une ensemble de taille respectable (toute la langue française s'il s'agit du dictionnaire) et voyons sur ce cas précis, facilement généralisable, les principales méthodes.

Recherche séquentielle

Aucune contrainte particulière : l'ensemble peut être quelconque et il n'est pas nécessaire de stocker d'autres informations que les mots eux-mêmes. Ne peut s'appliquer qu'à des ensembles de petite taille.

Recherche séquentielle indexée

Comme son nom l'indique cette technique fait appel à un index. On répartit les éléments de l'ensemble (les mots du dictionnaire) en plusieurs classes ou sous-ensembles. Chaque classe contiendra les mots commençant par la même lettre. On note ensuite dans un index les positions (par rapport au début du fichier) de chaque classe. Pour rechercher un mot on se positionne dans la classe correspondante (on connaît toutes les positions) et on entreprend dans la classe une recherche séquentielle. Cette technique permet donc de partitionner le domaine de recherche pour travailler sur un ensemble beaucoup plus petit.

Les contraintes sont ici de deux types :

- l'ensemble doit être découpé en plusieurs sous-ensembles. Il donc nécessaire de lui appliquer un certain traitement. Notons qu'au sein d'une classe les éléments ne sont pas ordonnés. De même les classes étant complétement indépendantes, elles peuvent apparaître dans n'importe quel ordre
- il est nécessaire de connaître les emplacements de toutes les classes donc de stocker une information supplémentaire.

Il est parfaitement possible d'augmenter le nombre de classes afin de réduire encore le temps d'acccès. Si l'on prend en compte les deux premières lettres on passe de 26 sous-ensembles à 676 (26^2). Les classes devenant de plus en plus petites les temps d'accés diminuent mais la quantité d'informations à stocker augmente.

On peut ainsi faire une partition de plus en plus fine et même travailler sur un ensemble dont chaque élément constitue une classe. L'accès à l'information est alors instantanné : on effectue de **l'accès direct** pur.

Remarquons que dans ce dernier cas il ne reste plus aucune contrainte sur l'ensemble puisque le travail de regroupement des éléments en classes est immédiat. Un singleton est en effet toujours 'regroupé' avec lui-même. Ce découpage présente donc un intérêt particulier.

Le nombre de positions à mémoriser étant très important (un entier par élément) il serait extrêmement intéressant de définir une fonction permettant à partir de l'élément de *calculer* sa position. Il ne serait alors plus nécessaire de mémoriser toutes ces positions. Cette technique s'appelle le **Hash-coding**.

Le Hash-coding

Sans trop entrer dans le détail décrivons le principe de cette méthode simple et pourtant peu connue. L'idée consiste donc à calculer la position de l'élément dans le fichier en lui appliquant une certaine fonction (par exemple la somme des positions dans l'alphabet des lettres composant le mot). Cette technique suppose que l'on dispose d'une fonction suffisamment sélective pour pouvoir qualifier différement et par là même pouvoir différencier chaque élément.

Ce n'est que très rarement le cas et plusieurs éléments peuvent alors fournir la même adresse de rangement, on parle de collisions. Il est malheureusement impossible de s'affranchir de cette limitation. On la contourne souvent en juxtaposant les éléments fournissant la même adresse. La recherche se fait alors en deux temps :

- calcul de la valeur à partir de l'élément et positionnement en accès direct sur l'enregistrement adéquat
- recherche séquentielle à partir de ce point d'entrée

Cette technique s'apparente donc fortement au séquentiel indexé. Bien qu'elle semble plus intéressante (rappelons qu'**aucune** information supplémentaire n'est à conserver), elle souffre, de par sa conception, d'une très importante restriction.

L'ensemble des adresses généré par la fonction n'est pratiquement jamais continu : c'est à dire qu'il peut comporter des trous. Ainsi, si l'ensemble des adresses est (2,11,3,20,7) on va utiliser 20 enregistrements (l'adresse la plus élevée) pour ranger 5 éléments. Certains systèmes d'exploitation réservent la place physique d'un enregistrement que s'il est non-vide. Cette discontinuité n'a alors aucune importance. Ce n'est hélas pas le cas de notre TI-99 qui réserve la place nécessaire à n enregistrements si on accède au $n^{ième}$ même s'ils sont tous vides....

L'accès direct en Pascal

Nous réalisons que toutes ces techniques utilisent abondamment le positionnement direct sur un enregistrement. L'instruction Pascal qui permet cela est **SEEK**(F,position). La position du premier enregistrement d'un fichier est zéro.

L'exécution de cette instruction provoque le déplacement de la fenêtre sur l'enregistrement spécifié. Il convient ensuite de faire GET(F) ou PUT(F) selon que l'on désire lire ou écrire.

Avec cette dernière commande nous disposons d'une panoplie d'instructions permettant de créer et mettre à jour les fichiers de tous poils.

Fichiers de types prédéfinis

Il existe deux principaux types de fichiers pré-établis. Ces fichiers peuvent être manipulés par d'autres instructions. Ils sont plus faciles à mettre à œuvre mais admettent certaines restrictions. Il s'agit des types **TEXT** et **INTERACTIVE**. Tous les deux sont composés de caractères. Les fichiers de ce type n'ont pas à être définis avec le mot-clé <u>file</u> mais simplement dans la rubrique <u>Var</u> comme une variable classique.

Ainsi

..........

```
var            f : text;
```

indique au programme que f est un fichier de caractères exactement comme

```
var            f : file of char;
```

l'aurait fait.

Les fichiers de caractères ont une structure particulière. Ils sont formés d'enregistrements (eux-mêmes composés de caractères) de taille différente. Chaque enregistrement (ligne de texte) se termine par un caractère spécial qui marque la fin de la ligne. La présence de cette marque peut être testée par la fonction booléenne **EOLN(F)** qui est vraie si la fin de la ligne a été atteinte et fausse autrement.

La lecture (respectivement l'écriture) sur un tel fichier se fait par **READ**(F,variable_lue) et **READLN** (respectivement **WRITE** ou **WRITELN**). Ces instructions ont pour premier argument la variable fichier (celle utilisée lors du RESET ou du REWRITE) suivi de la (ou des) variable(s) à lire (ou à écrire). READ (WRITE) ne lit (écrit) qu'un seul caractère alors que READLN (WRITELN) manipule une ligne entière.

Si F est de type TEXT et CH de type CHAR READ(F,CH) lit le caractère situé sous la fenêtre. Cette instruction est donc équivalente à

```
CH=F^;
GET(F);
```

alors que pour un fichier de type INTERACTIVE l'ordre de ces instructions est inversé ainsi READ(F,CH) est équivalent à

```
GET(F);
CH=F^;
```

Cette différence peut paraître minime, pourtant elle a son importance. Par exemple si on teste la fin de la ligne au moyen de EOLN(F), pour un fichier de type INTERACTIVE EOLN ne devient vraie qu'**après** avoir lu le caractère qui marque la fin de la ligne.

Cette marque est donc lue avec un fichier INTERACTIVE, ce n'est pas le cas avec un fichier TEXT.

Les fichiers créés à partir de l'éditeur de texte du système peuvent être lus (et modifiés) en les considérant comme TEXT ou INTERACTIVE.

La principale limitation des ces fichiers est inhérente à la structure même de leurs enregistrements. Comme ils ont une taille variable (la marque de fin de ligne peut se trouver n'importe où) le système est incapable de calculer l'emplacement physique d'un enregistrement donné sans parcourir le fichier jusqu'à lui. L'accès direct n'est donc pas envisageable. Il est néanmoins possible de faire de l'accès direct sur un fichier de caractères en déclarant explicitement sa structure, les enregistrements auront alors une taille fixe. Ainsi :

```
type blabla = packed record of
                ligne : string[80]
                end;

var fichier : file of blabla;
```

ou directement :

```
   var fichier : file of string[80];
```

permet d'utiliser l'instruction SEEK.

Programme exemple

Le petit programme suivant illustre certaines des commandes décrites auparavant. Le fichier ADRESSES.DATA regroupe les informations de toutes les personnes.

La technique employée ici est l'accès direct simple pour *chaque* enregistrement. En début d'exécution ce fichier est intégralement parcouru et un *tableau-index* est créé. Ce tableau regroupe les cinq premières lettres du champ 'nom' de chaque enregistrement du fichier.

Ainsi pour accéder aux informations relatives à une personne il faut donner au programme son nom. Il cherche alors dans l'index la position de l'enregistrement correspondant à cette personne.
Notez dans le listing le présence de la directive {$I-} qui, associée à **IORESULT**, permet de tester la présence de ADRESSES.DAT. Si ce fichier est absent, RESET provoque une erreur : on emploie alors REWRITE. On crée ensuite le premier enregistrement du fichier, on le ferme en le sauvegardant, puis on l'ouvre à nouveau avec RESET qui permet aussi bien de lire que d'écrire.

De conception très simple ce programme gère directement le fichier sur disquette. Les opérations de lecture ou d'écriture sont évidemment plus longues (l'écriture sur fichier est beaucoup moins rapide que l'affectation d'une variable) mais le programme peut ainsi appréhender des quantités d'informations qui ne "tiendraient" pas en mémoire centrale.

A vous d'enrichir ce programme en lui adjoignant, par exemple, des algorithmes de tri, d'édition sélective ou de sélection multi-critères. Faites nous part de vos suggestions... □

```
program FICHES;

const bidon=false;

var f        : file of record
           nom     : string[20];
           prenom  : string[15];
           tel_dom : string[13];
           tel_bur : string[22];
           adr1    : string[25];
           adr2    : string[20];
           adr3    : string[10];
           code    : string[6];
           ville   : string[20];
           end;

    choix,nmax   : integer;
    pers    : packed array[1..50] of string[5];

procedure poursuivre;
begin
gotoxy(15,23);
write('ENTER pour continuer',chr(7));
readln
end;


procedure lister;
var i :integer;
begin
page(output);
writeln('L I S T E   D E S   N O M S');
writeln('---------------------------');
gotoxy(0,5);
seek(f,0);
for i:=1 to nmax do
    begin
    get(f);
    writeln('':5,f^.nom)
    end;
for i:=1 to 6 do writeln;
poursuivre
end;


procedure consulter;
var i    : integer;
    nom1: string[5];
    nom2: string[20];
begin
page(output);
writeln('C O N S U L T E R   U N E   F I C H E');
writeln('---------------------------------------');
gotoxy(0,8);
write('Nom       : ');
readln(nom2);
if length(nom2)<5 then
   nom2:=concat(nom2,'    ');
nom1:=copy(nom2,1,5);
i:=0;
  repeat
  i:=i+1
  until (pers[i]=nom1) or (i=nmax+1);
if i=nmax+1 then
   begin
   gotoxy(15,19);
   writeln('non repertorie ...');
   end
else
   begin
   seek(f,i-1);
   get(f);
   writeln('':10,f^.prenom);
   writeln('Tel dom : ',f^.tel_dom);
   writeln('Tel bur : ',f^.tel_bur);
   writeln('':10,f^.adr1);
   writeln('':10,f^.adr2);
   writeln('':10,f^.adr3);
   writeln('':10,f^.code,'':3,f^.ville);
   end;
poursuivre
end;


procedure modifier;
var nom1: string[5];
    nom2: string[20];
    i    : integer;
begin
page(output);
writeln('M O D I F I E R   U N E   F I C H E');
writeln('-----------------------------------');
gotoxy(0,5);
write('Nom      : ');
readln(nom2);
if length(nom2)<5 then
   nom2:=concat(nom2,'    ');
nom1:=copy(nom2,1,5);
i:=0;
  repeat
  i:=i+1
  until (pers[i]=nom1) or (i=nmax+1);
if i=nmax+1 then
   begin
   gotoxy(15,19);
   writeln('non repertorie    ');
   end
else
   begin
   seek(f,i-1);
   get(f);
   writeln('':10,f^.prenom);
   writeln('Tel bur : ',f^.tel_bur);
   write('nou.val.: ');
   readln(f^.tel_bur);
   writeln('Tel dom . '.f^.tel_dom);
   write('nou.val.: ');
   readln(f^.tel_dom);
   writeln('Adr1    : ',f^.adr1);
   write('nou.val.: ');
   readln(f^.adr1);
```

```
    writeln('Adr2    : ',f^.adr2);
    write('nou.val.: ');
    readln(f^.adr2);
    writeln('Adr3    : ',f^.adr3);
    write('nou.val.: ');
    readln(f^.adr3);
    writeln('Code    : ',f^.code);
    write('nou.val.: ');
    readln(f^.code);
    writeln('Ville   : ',f^.ville);
    write('nou.val.: ');
    readln(f^.ville);
    seek(f,i-1);
    put(f);
    end;
 poursuivre
 end;


 procedure supprimer;
 var nom1: string[5];
     nom2: string[20];
     i,j : integer;
 begin
 page(output);
 writeln('S U P P R I M E R   U N E   F I C H E');
 writeln('--------------------------------------');
 gotoxy(0,8);
 write('Nom      : ');
 readln(nom2);
 if length(nom2)<5 then
    nom2:=concat(nom2,'     ');
 nom1:=copy(nom2,1,5);
 i:=0;
   repeat
   i:=i+1
   until (pers[i]=nom1) or (i=nmax+1);
 if i=nmax+1 then
    begin
    gotoxy(15,19);
    writeln('non repertorie ...');
    end
 else
    begin
    for j:=i+1 to nmax do
        begin
        pers[j-1]:=pers[j];
        seek(f,j-1);
        get(f);
        seek(f,j-2);
        put(f);
        end;
    nmax:=nmax-1
    end;
 poursuivre
 end;


 procedure ajouter;
 begin
 page(output);
 writeln('A J O U T E R   U N E   F I C H E');
 writeln('----------------------------------');

 seek(f,nmax);
 gotoxy(0,8);
 write('Nom      : '); readln(f^.nom);
 if length(f^.nom)<5 then
    f^.nom:=concat(f^.nom,'     ');
 write('Prenom   : '); readln(f^.prenom);
 write('Tel dom  : '); readln(f^.tel_dom);
 write('Tel bur  : '); readln(f^.tel_bur);
 write('Adr. 1/3: '); readln(f^.adr1);
 write('Adr. 2/3: '); readln(f^.adr2);
 write('Adr. 3/3: '); readln(f^.adr3);
 write('Code     : '); readln(f^.code);
 write('Ville    : '); readln(f^.ville);
 nmax:=nmax+1;
 pers[nmax]:=copy(f^.nom,1,5);
 put(f)
 end;


 procedure logo;
 begin
 page(output);
 writeln('':5,'Gestion d''un fichier d''adresses');
 writeln('(C)opyright 99 Magazine et G. SANTRAILLE');
 gotoxy(10,10)
 end;



 begin     (debut du programme principal)
 ($I-)
 reset(f,'ADRESSES.DAT');
 ($i+)
 if ioresult<>0 then
    begin
    page(output);
    nmax:=0;
    writeln('ADRESSES.DAT n''existe pas');
    writeln('definition du premier enregistrement');
    rewrite(f,'ADRESSES.DAT');
    poursuivre;
    ajouter;
    close(f,lock);
    reset(f,'ADRESSES.DAT');
    end;
 logo;
 gotoxy(15,10);
 writeln('Un peu de patience');

    nmax:=0;
    while not eof(f) do
    begin
    nmax:=nmax+1;
    pers[nmax]:=copy(f^.nom,1,5);
    get(f)
    end;


 repeat
   logo;
   writeln('1 -> Liste des noms');
   writeln('':10,'2 -> Consulter une fiche');
   writeln('':10,'3 -> Modifier une fiche');
   writeln('':10,'4 -> Supprimer une fiche');
   writeln('':10,'5 -> Ajouter une fiche');
   writeln('':10,'6 -> Finir');
   writeln;
   writeln('':20,'Votre choix ?');
       repeat
       writeln(chr(7));
       gotoxy(34,17);
       readln(choix);
       until (choix<7) and (choix>0);
   case choix of
     1 : lister;
     2 : consulter;
     3 : modifier;
     4 : supprimer;
     5 : ajouter;
     6 : begin
           page(output);
           seek(f,nmax);
           close(f,crunch);
           exit(program)
         end
   end;
 until bidon=true
 end.
```

Editeur de caractères et lutins

Julien Thomas

Avec ce programme, écrit en **Basic Etendu**, il est possible de se passer de l'éternelle feuille de papier quadrillée pour obtenir les codes nécessaires à la redéfinition d'un caractère ou d'un lutin.

L'écran du TI affiche une grille de 16 * 16 carrés, représentant 256 pixels; il est donc possible de définir simultanément quatre caractères graphiques. Pour faire passer les pixels de l'état "off" à l'état "on", il suffit de ce déplacer dans la grille à l'aide des joysticks. Pour faire passer les points de "on' à "off", il faut appuyer sur le "bouton de tir".

Pour mieux juger de l'effet de votre création, le programme affiche le ou les caractères à leur taille réelle et il est possible de changer la couleur des points "on" et "off".

Vous pouvez obtenir l'affichage des codes hexadécimaux à l'écran ou sur imprimante. Pour le dernier cas, vous obtenez la représentation du caractère sur le listing; la transmission à l'imprimante ne se fait pas en haute résolution, ce qui autorise une adaptation aisée à différents types d'imprimante. Le programme est prévu pour une imprimante parallèle; pour une imprimante série, il faut modifier la ligne 800 (OPEN#1:"RS232. BA=9600", par exemple).

□

```
10 CALL CLEAR :: CALL DELSPRITE(#1):: RE
STORE :: DIM BI(4),HD$(15),CI$(16):: BI(
1)=8 :: BI(2)=4 :: BI(3)=2 :: BI(4)=1
20 FOR B=0 TO 9 :: HD$(B)=STR$(B):: NEXT
 B :: FOR B=10 TO 15 :: HD$(B)=CHR$(B+55
):: NEXT B
30 DATA TRANSPA,NOIR,VERT M,VERT C,BLEU
F,BLEU C,ROUGE F,CYAN,ROUGE M,ROUGE C,JA
UNE F,JAUNE C,VERT F,MAGENTA,GRIS,BLANC
100 A$="FF80808080808080" :: B$=RPT$("F",
16):: CALL CHAR(96,A$,97,RPT$("80",8),98
,"FF",99,B$,104,A$,112,B$):: CALL COLOR(
10,2,9)
110 CALL HCHAR(17,3,98,16):: FOR B=1 TO
16 :: CALL HCHAR(B,3,96,16):: NEXT B ::
CALL VCHAR(1,19,97,16):: CALL HCHAR(8,8,
104):: PH,PV=8
120 CALL JOYST(1,X,Y):: CALL KEY(1,K,S):
: IF K=18 THEN C=0 :: CALL COLOR(10,2,11
)ELSE C=1 :: CALL COLOR(10,2,9)
130 IF K=19 THEN CALL HCHAR(PV,PH,99)::
GOTO 500
140 IF X=0 AND Y=4 THEN 200 ELSE IF X=4
AND Y=0 THEN 250 ELSE IF X=0 AND Y=-4 TH
EN 300 ELSE IF X=-4 AND Y=0 THEN 350 ELS
E 120
200 IF PV=1 THEN 120 ELSE PV=PV-1 :: GOS
UB 2000
210 CALL HCHAR(PV+1,PH,CA):: GOTO 400
250 IF PH=18 THEN 120 ELSE PH=PH+1 :: GO
SUB 2000
260 CALL HCHAR(PV,PH-1,CA):: GOTO 400
300 IF PV=16 THEN 120 ELSE PV=PV+1 :: GO
SUB 2000
310 CALL HCHAR(PV-1,PH,CA):: GOTO 400
350 IF PH=3 THEN 120 ELSE PH=PH-1 :: GOS
UB 2000
360 CALL HCHAR(PV,PH+1,CA)
400 FOR B=0 TO 20 :: NEXT B :: GOTO 120
500 FOR B1=1 TO 8 :: FOR B2=3 TO 6 :: CA
LL GCHAR(B1,B2,COC):: IF COC=99 THEN COM
P=COMP+BI(B2-2)
510 NEXT B2 :: GOSUB 2100
520 FOR B3=7 TO 10 :: CALL GCHAR(B1,B3,C
OC):: IF COC=99 THEN COMP=COMP+BI(B3-6)
525 NEXT B3 :: GOSUB 2100 :: NEXT B1
530 FOR B4=9 TO 16 :: FOR B5=3 TO 6 :: C
ALL GCHAR(B4,B5,COC):: IF COC=99 THEN CO
MP=COMP+BI(B5-2)
540 NEXT B5 :: GOSUB 2100
550 FOR B6=7 TO 10 :: CALL GCHAR(B4,B6,C
OC):: IF COC=99 THEN COMP=COMP+BI(B6-6)
560 NEXT B6 :: GOSUB 2100 :: NEXT B4
570 FOR B7=1 TO 8 :: FOR B8=11 TO 14 ::
CALL GCHAR(B7,B8,COC):: IF COC=99 THEN C
OMP=COMP+BI(B8-10)
575 NEXT B8 :: GOSUB 2100
580 FOR B9=15 TO 18 :: CALL GCHAR(B7,B9,
COC):: IF COC=99 THEN COMP=COMP+BI(B9-14
)
590 NEXT B9 :: GOSUB 2100 :: NEXT B7
600 FOR B10=9 TO 16 :: FOR B11=11 TO 14
:: CALL GCHAR(B10,B11,COC):: IF COC=99 T
HEN COMP=COMP+BI(B11-10)
610 NEXT B11 :: GOSUB 2100
620 FOR B12=15 TO 18 :: CALL GCHAR(B10,B
12,COC):: IF COC=99 THEN COMP=COMP+BI(B1
2-14)
630 NEXT B12 :: GOSUB 2100 :: NEXT B10
700 DISPLAY AT(17,1):"CODE HEXADECIMAL:"
 :: DISPLAY AT(18,1):CHAINE$
710 RESTORE :: CALL DELSPRITE(#1):: FOR
B=1 TO 16 :: DISPLAY AT(B,18):USING "##"
:B :: READ COU$ :: DISPLAY AT(B,21):COU$
```

```
 :: NEXT B
720 DISPLAY AT(22,1):"" :: DISPLAY AT(23
,1):"COULEUR DU DESSIN?" :: ACCEPT AT(23
,20)VALIDATE(DIGIT):CODE :: IF CODE<1 OR
 CODE>16 THEN 720
730 DISPLAY AT(24,1):"COULEUR DE FOND?"
:: ACCEPT AT(24,18)VALIDATE(DIGIT):COFO
:: IF COFO<1 OR COFO>16 THEN 730
740 FOR B=1 TO 16 :: DISPLAY AT(B,18):""
 :: NEXT B :: CALL COLOR(11,COFO,1)
750 FOR B=1 TO 6 :: CALL HCHAR(B,22,112,
6):: NEXT B :: CALL CHAR(120,CHAINE$)::
CALL MAGNIFY(3)
760 CALL SPRITE(#1,120,CODE,17,185)
770 DISPLAY AT(21,1):"1-MODIFICATION COU
LEURS" :: DISPLAY AT(22,1):"2-MODIFICATI
ON DESSIN" :: DISPLAY AT(23,1):"3-NOUVEA
U DESSIN"
775 DISPLAY AT(24,1):"4-IMPRESSION
    5-FIN"
780 CALL KEY(1,K1,S1):: IF S1=0 THEN 780
785 FOR B=21 TO 24 :: DISPLAY AT(B,1):""
 :: NEXT B
786 IF K1=9 THEN 800 ELSE IF K1=10 THEN
CALL DELSPRITE(#1):: CALL CLEAR :: END
790 IF K1=19 THEN 710 ELSE IF K1=7 THEN
CHAINE$="" :: CALL HCHAR(PV,PH,104):: GO
TO 120 ELSE IF K1=8 THEN CHAINE$="" :: G
OTO 10 ELSE 770
800 OPEN #1:"PIO.LF"
810 FOR BL=1 TO 16 :: FOR BC=3 TO 18 ::
CALL GCHAR(BL,BC,CI)
820 IF CI=99 THEN CI$(BL)=CI$(BL)&"#" EL
SE CI$(BL)=CI$(BL)&" "
830 NEXT BC :: NEXT BL
840 FOR BIM=1 TO 16 :: PRINT #1:CHR$(14)
;TAB(5);CI$(BIM):: NEXT BIM
870 PRINT #1:CHR$(15);"":"CODE HEXADECIM
AL":CHAINE$
880 CLOSE #1 :: FOR BEF=1 TO 16 :: CI$(B
EF)="" :: NEXT BEF :: GOTO 770
2000 CALL HCHAR(PV,PH,104):: IF C THEN C
A=99 ELSE CA=96
2010 RETURN
2100 CHAINE$=CHAINE$&HD$(COMP):: COMP=0
:: RETURN
```

Kaleïdoscope

Marianne Sutz

Ce petit programme en **Basic Etendu** se passe de commentaires, et prouve, si besoin était, que le TI est un très bon outil graphique puisqu'il permet d'obtenir des effets extrêmement intéressants avec seulement quelques lignes de Basic.

□

```
100 CALL SCREEN(1):: CALL CLEAR :: RANDO
MIZE :: OPTION BASE 1 :: DIM COO(14),COF
(14)
110 FOR B=33 TO 137 STEP 8 :: CALL CHAR(
B,"0F0F0F0FF0F0F0F0",B+1,"F0F0F0F00F0F0F
0F",B+2,"0F0F0F0FF0F0F0F0"):: NEXT B
120 FOR B=1 TO 14 :: CALL COLOR(B,16,1):
: COF(B),COO(B)=1 :: NEXT B
150 FOR L=1 TO 6 :: FOR C=1 TO 8 :: CALL
 HCHAR(L,C,(25+(INT(RND*14)+1)*8)+INT(RN
D*2)):: NEXT C :: NEXT L
160 FOR L=1 TO 6 :: FOR C=1 TO 8 :: CALL
 GCHAR(L,C,X):: CALL HCHAR(L,17-C,X+1)::
 CALL HCHAR(L,16+C,X):: CALL HCHAR(L,33-
C,X+1)
161 CALL HCHAR(13-L,C,X+1):: CALL HCHAR(
13-L,17-C,X):: CALL HCHAR(13-L,16+C,X+1)
:: CALL HCHAR(13-L,33-C,X):: CALL HCHAR(
12+L,C,X)
162 CALL HCHAR(12+L,17-C,X+1):: CALL HCH
AR(12+L,16+C,X):: CALL HCHAR(12+L,33-C,X
+1):: CALL HCHAR(25-L,C,X+1)
163 CALL HCHAR(25-L,17-C,X):: CALL HCHAR
(25-L,16+C,X+1):: CALL HCHAR(25-L,33-C,X
):: NEXT C :: NEXT L
170 CALL KEY(0,K,S):: IF S THEN CALL CLE
AR :: GOTO 150 ELSE CAR=INT(RND*14)+1 ::
 COO(CAR)=INT(RND*16)+1 :: CALL COLOR(CA
R,COO(CAR),COF(CAR))
180 CALL KEY(0,K,S):: IF S THEN CALL CLE
AR :: GOTO 150 ELSE CAR=INT(RND*14)+1 ::
 COF(CAR)=INT(RND*16)+1 :: CALL COLOR(CA
R,COO(CAR),COF(CAR))
185 GOTO 170
```

Opérations

Albert Rigal

Ce programme éducatif, écrit en **Basic TI**, permet de faire réciter les tables des quatres opérations à de jeunes écoliers.

Il propose :

- le choix de l'opération;
- le choix de la table;
- une série de dix questions dans un ordre aléatoire;
- si deux erreur à une même question : affichage de la bonne réponse;
- trois niveaux de notation avec appréciations à chaque niveau;
- selon la note attribuée, l'affichage automatique de la table pour faciliter la révision.

Mode d'emploi

Après le générique d'introduction, le programme commence par demander le prénom de l'enfant, ce qui permettra de personnaliser le programme. Ensuite, le menu s'affiche, on choisit l'opération (multiplication, soustraction, addition et division) puis la table.

Alors, une série de dix questions est posée, ce qui permet de faire réciter toute la table dans un ordre aléatoire. On entre la réponse, puis <ENTER>.

A la fin de la série, la note s'affiche, ainsi qu'une appréciation. Une note égale ou inférieure à 6/10 provoque l'affichage de la table en question, pour la révision tout le temps désiré. On répond ensuite à la question "Une autre série ? O/N".

```
100 REM*******************

110 REM*    OPERATIONS    *

120 REM*   ALBERT RIGAL   *

130 REM*******02.1984******

140 REM*  POUR TI 99/4A   *

150 REM*     TI BASIC     *

160 REM*******************
170 GOSUB 3070
180 CALL CLEAR
190 RANDOMIZE
200 CALL SCREEN(7)
210 FOR I=5 TO 8
220 CALL COLOR(I,4,1)
230 NEXT I
240 FOR I=1 TO 4
250 CALL COLOR(I,16,1)
260 NEXT I
270 CALL CLEAR
280 INPUT "QUEL EST TON PRENOM

        S'IL TE PLAIT
 ?  ":P$
290 CALL CLEAR
300 PRINT "MAINTENANT ";P$;"

        AU TRAVAIL!!.
..": : : : : : : : : :
310 CALL SOUND(100,800,0)
320 FOR I=1 TO 600
330 NEXT I
340 REM ROUTINE TRANSPARENCE

350 CALL CLEAR
360 FOR I=1 TO 8
370 CALL COLOR(I,1,1)
380 NEXT I
390 REM MENU
400 PRINT "QUELLE OPERATION CH
OISIS-TU?": :
410 PRINT "..................
........": : : :
420 PRINT "      MULTIPLICATION
 :1": :
430 PRINT "         SOUSTRACTION
 :2": :
440 PRINT "             ADDITION
 :3": :
450 PRINT "             DIVISION
 :4": : : :
460 PRINT "..................
........": : : : : :"     TAPE
UN CHIFFRE DE 1 A 4"
470 REM COULEURS CARACTERES
480 FOR I=5 TO 8
490 CALL COLOR(I,4,1)
500 NEXT I
510 FOR I=1 TO 4
520 CALL COLOR(I,16,1)
530 NEXT I
540 CALL KEY(0,K,S)
550 IF (S=0)+(K<49)+(K>52)THEN
 540
560 K=K-48
570 ON K GOSUB 590,1200,1810,2
420
580 GOTO 350
590 CALL CLEAR
600 REM MULTIPLICATION
610 PRINT "QUELLE TABLE

        DE MULTIPLICA
TION
               CHOISIS-TU ? DE 1
 A 9": : : :
620 PRINT "            tape u
n chiffre"
630 CALL KEY(5,K,S)
640 IF (S=0)+(K<49)+(K>57)THEN
 630
650 A=K-48
660 CALL CLEAR
670 X=0
680 DIM N(10)
690 FOR I=1 TO 10
700 N(I)=I
710 NEXT I
720 FOR I=10 TO 1 STEP -1
730 R=INT(RND*I)+1
740 Q=A*N(R)
750 INPUT STR$(A)&" FOIS "&STR
$(N(R))&" EGAL? ":T
760 IF T<>Q THEN 800
770 PRINT :"
  JUSTE...": :
780 Z=0
790 GOTO 880
800 Z=Z+1
810 IF Z=2 THEN 3800
820 PRINT :"      ERREUR! ";P$
;: :"                RECOMMENCE
...": :
830 X=X+1
840 CALL SOUND(500,200,0)
850 FOR V=1 TO 300
860 NEXT V
870 GOTO 750
880 IF R=I THEN 920
890 FOR J=R+1 TO I
900 N(J-1)=N(J)
910 NEXT J
920 NEXT I
930 Y=10-X
940 IF Y>=9 THEN 970
950 IF (Y=8)+(Y=7)THEN 1030
960 IF Y<=6 THEN 1070
970 PRINT :"------------------
----------": :"TRES BIEN ";P$;
"
                             BRA
VO!!!": :
980 CALL SOUND(200,400,1)
990 CALL SOUND(200,500,1)
1000 CALL SOUND(200,600,1)
1010 CALL SOUND(200,800,1)
1020 GOTO 1090
1030 PRINT :"-----------------
-----------": :"BIEN !. ";P$;"
": :
1040 CALL SOUND(200,600,1)
1050 CALL SOUND(200,800,1)
1060 GOTO 1090
1070 PRINT :"-----------------
-----------": :"PAS FAMEUX !.
";P$;"
           FAUDRAS REVISER
```

```
LA TABLE...": : :
1080 CALL SOUND(200,800,1)
1090 PRINT :"RESULTATS:";Y;"SU
R 10...": :"------------------
----------": :
1100 FOR E=1 TO 800
1110 NEXT E
1120 IF Y>=7 THEN 1140
1130 GOSUB 3680
1140 PRINT :"UNE AUTRE SERIE ?
 O/N": :
1150 CALL KEY(0,K,S)
1160 IF (S=0)+(K<78)+(K>79)THE
N 1150
1170 IF K=78 THEN 3020
1180 RETURN
1190 REM SOUSTRACTION
1200 CALL CLEAR
1210 CALL SCREEN(14)
1220 PRINT :"QUELLE TABLE DE S
OUSTRACTION
        CHOISIS-TU
? DE 1 A 9": : : :
1230 PRINT "                tape
un chiffre"
1240 CALL KEY(5,K,S)
1250 IF (S=0)+(K<49)+(K>57)THE
N 1240
1260 A=K-48
1270 CALL CLEAR
1280 X=0
1290 FOR I=1 TO 10
1300 N(I)=I
1310 NEXT I
1320 FOR I=10 TO 1 STEP -1
1330 Z=0
1340 R=INT(RND*I)+1
1350 INPUT STR$(N(R)+10)&" MOI
NS "&STR$(A)&" EGAL? ":T
1360 Q=N(R)+10-A
1370 IF T<>Q THEN 1410
1380 PRINT :"
    VRAI..."
1390 Z=0
1400 GOTO 1490
1410 Z=Z+1
1420 IF Z=2 THEN 3950
1430 PRINT :"        FAUX ! ";P
$;: :"               RECOMMENC
E...": :
1440 X=X+1
1450 CALL SOUND(500,200,0)
1460 FOR V=1 TO 300
1470 NEXT V
1480 GOTO 1350
1490 IF R=I THEN 1530
1500 FOR J=R+1 TO I
1510 N(J-1)=N(J)
1520 NEXT J
1530 NEXT I
1540 Y=10-X
1550 IF Y>=9 THEN 1580
1560 IF (Y=8)+(Y=7)THEN 1640
1570 IF Y<=6 THEN 1680
1580 PRINT :"-----------------
-----------": :"BRAVO!! ";P$;"
 TRES BIEN..": :
1590 CALL SOUND(200,400,1)
1600 CALL SOUND(200,500,1)
1610 CALL SOUND(200,600,1)
1620 CALL SOUND(200,800,1)
1630 GOTO 1700
1640 PRINT :"-----------------
-----------": :"BIEN!!..";P$;:
 :
1650 CALL SOUND(200,600,1)
1660 CALL SOUND(200,800,1)
1670 GOTO 1700
1680 PRINT :"-----------------
-----------": :"MEDIOCRE !!. "
;P$;"
             ETUDIE LA TABLE.
..": :
1690 CALL SOUND(200,800,1)
1700 PRINT :"RESULTATS :";Y;"S
UR 10...": :"-----------------
-----------":;
1710 FOR E=1 TO 800
1720 NEXT E
1730 IF Y>=7 THEN 1750
1740 GOSUB 3830
1750 PRINT :"UNE AUTRE SERIE?
O/N"
1760 CALL KEY(5,K,S)
1770 IF (S=0)+(K<78)*(K>79)THE
N 1760
1780 IF K=78 THEN 3020
1790 RETURN
1800 REM *ADDITION*********
1810 CALL CLEAR
1820 CALL SCREEN(6)
1830 PRINT :"QUELLE TABLE D'AD
DITION
          CHOISIS-TU
? DE 1 A 9 ": : : :
1840 PRINT "                tape
un chiffre"
1850 CALL KEY(5,K,S)
1860 IF (S=0)+(K<49)+(K>57)THE
N 1850
1870 A=K-48
1880 CALL CLEAR
1890 X=0
1900 FOR I=1 TO 10
1910 N(I)=I
1920 NEXT I
1930 FOR I=10 TO 1 STEP -1
1940 Z=0
1950 R=INT(RND*I)+1
1960 Q=A+N(R)
1970 INPUT STR$(A)&" PLUS "&ST
R$(N(R))&" EGAL? ":T
1980 IF T<>Q THEN 2020
1990 PRINT :"
   EXACT..."
2000 Z=0
2010 GOTO 2110
2020 Z=Z+1
2030 IF Z=2 THEN 4100
2040 PRINT :"           NON! ";
P$;: :"                RECOMMEN
CE...": :
2050 X=X+1
2060 CALL SOUND(500,200,0)
2070 FOR V=1 TO 300
2080 NEXT V
2090 GOTO 1970
2100 IF R=I THEN 2140
2110 FOR J=R+1 TO I
2120 N(J-1)=N(J)
2130 NEXT J
2140 NEXT I
2150 Y=10-X
2160 IF Y>=9 THEN 2190
2170 IF (Y=8)+(Y=7)THEN 2250
2180 IF Y<=6 THEN 2290
2190 PRINT :"-----------------
-----------": :"C'EST PARFAIT
";P$;"
                            BR
AVO !!!..": :
2200 CALL SOUND(200,400,1)
2210 CALL SOUND(200,500,1)
2220 CALL SOUND(200,600,1)
2230 CALL SOUND(200,800,1)
2240 GOTO 2310
2250 PRINT :"-----------------
-----------": :"BIEN..";P$;"":
 :
2260 CALL SOUND(200,600,1)
2270 CALL SOUND(200,800,1)
2280 GOTO 2310
2290 PRINT :"-----------------
-----------": :"PAS BON DU TOU
T !!..";P$;: :"
          ETUDIE...": :
2300 CALL SOUND(200,800,1)
2310 PRINT :"RESULTATS :";Y;"S
UR 10...": :"-----------------
-----------"
2320 FOR E=1 TO 800
2330 NEXT E
2340 IF Y>=7 THEN 2360
2350 GOSUB 3980
2360 PRINT :"UNE AUTRE SERIE ?
 O/N": :
2370 CALL KEY(5,K,S)
2380 IF (S=0)+(K<78)+(K>79)THE
N 2370
2390 IF K=78 THEN 3020
2400 RETURN
2410 REM DIVISION
2420 CALL CLEAR
2430 CALL SCREEN(7)
2440 PRINT :"QUELLE TABLE DE D
IVISION
        CHOISIS-TU
? DE 1 A 9": : : :
2450 PRINT "                tape
un chiffre"
2460 CALL KEY(5,K,S)
2470 IF (S=0)+(K<49)+(K>77)THE
N 2460
2480 CALL CLEAR
2490 A=K-48
2500 X=0
2510 FOR I=1 TO 10
2520 N(I)=I
2530 NEXT I
2540 FOR I=10 TO 1 STEP -1
2550 Z=0
2560 R=INT(RND*I)+1
2570 Q=N(R)*A/A
2580 INPUT STR$(N(R)*A)&" QUE
DIVISE "&STR$(A)&" EGAL ":T
2590 IF T<>Q THEN 2630
2600 PRINT :"
    BON...": :
2610 Z=0
2620 GOTO 2710
2630 Z=Z+1
2640 IF Z=2 THEN 4250
2650 PRINT :"    FAUX !. ";P$;
: :"              RECOMMENCE.
..": :
2660 X=X+1
2670 CALL SOUND(500,200,0)
2680 FOR V=1 TO 300
2690 NEXT V
2700 GOTO 2580
2710 IF R=I THEN 2750
2720 FOR J=R+1 TO I
2730 N(J-1)=N(J)
2740 NEXT J
2750 NEXT I
2760 Y=10-X
2770 IF Y>=9 THEN 2800
2780 IF (Y=8)+(Y=7)THEN 2860
2790 IF Y<=6 THEN 2900
2800 PRINT :"-----------------
-----------": :"TRES BIEN ";P$
```

```
;" BRAVO !!!.": :
2810 CALL SOUND(200,400,1)
2820 CALL SOUND(200,500,1)
2830 CALL SOUND(200,600,1)
2840 CALL SOUND(200,800,1)
2850 GOTO 2920
2860 PRINT :"------------------
-----------": :"C'EST BIEN !!"
;P$;"": :
2870 CALL SOUND(200,600,1)
2880 CALL SOUND(200,800,1)
2890 GOTO 2920
2900 PRINT :"------------------
-----------": :"PAS FAMEUX !!.
. ";P$;"
                    TU NE SAIS PAS
 LA TABLE...": :
2910 CALL SOUND(200,800,1)
2920 PRINT :"RESULTATS :";Y;"S
UR 10...": :"------------------
-----------"
2930 FOR E=1 TO 800
2940 NEXT E
2950 IF Y>=7 THEN 2970
2960 GOSUB 4130
2970 PRINT :"UNE AUTRE SERIE ?
 O/N"
2980 CALL KEY(5,K,S)
2990 IF (S=0)+(K<78)+(K>79)THE
N 2980
3000 IF K=78 THEN 3020
3010 RETURN
3020 CALL CLEAR
3030 GOSUB 4290
3040 CALL CLEAR
3050 END
3060 REM AFFICHAGE GENERIQUE
3070 CALL CLEAR
3080 CALL SCREEN(16)
3090 CALL CHAR(90,"FFFFFFFFFF
FFFFF")
3100 CALL CHAR(130,"FFFFFFFFFF
FFFFFF")
3110 CALL CHAR(136,"FFE7C38181
C3E7FF")
3120 FOR T=1 TO 14
3130 CALL COLOR(T,16,1)
3140 NEXT T
3150 CALL HCHAR(4,9,90,19)
3160 CALL HCHAR(5,10,136,17)
3170 CALL VCHAR(6,10,136,13)
3180 CALL VCHAR(5,9,90,15)
3190 CALL VCHAR(5,27,90,15)
3200 CALL VCHAR(6,26,136,13)
3210 CALL HCHAR(6,11,130,15)
3220 CALL HCHAR(7,11,130,15)
3230 CALL HCHAR(8,11,130,15)
3240 CALL HCHAR(9,11,130,15)
3250 CALL HCHAR(10,11,130,15)
3260 CALL HCHAR(11,11,130,15)
3270 CALL HCHAR(12,11,136,15)
3280 CALL HCHAR(13,11,130,15)
3290 CALL HCHAR(14,11,130,15)
3300 CALL HCHAR(15,11,130,15)
3310 CALL HCHAR(16,11,130,15)
3320 CALL HCHAR(17,11,130,15)
3330 CALL HCHAR(18,11,130,15)
3340 CALL HCHAR(19,10,136,17)
3350 CALL HCHAR(20,9,90,19)
3360 A$="PROGRAMME"
3370 B$="-OPERATIONS-"
3380 C$=" * / + - "
3390 E$="02.1984"
3400 FOR I=1 TO LEN(A$)
3410 CALL HCHAR(7,13+I,ASC(SEG
$(A$,I,1)))
3420 NEXT I
3430 FOR I=1 TO LEN(B$)
3440 CALL HCHAR(10,11+I,ASC(SE
G$(B$,I,1)))
3450 NEXT I
3460 FOR I=1 TO LEN(C$)
3470 CALL HCHAR(14,13+I,ASC(SE
G$(C$,I,1)))
3480 NEXT I
3490 FOR I=1 TO LEN(E$)
3500 CALL HCHAR(17,14+I,ASC(SE
G$(E$,I,1)))
3510 NEXT I
3520 CALL SCREEN(2)
3530 CALL COLOR(8,5,1)
3540 FOR R=1 TO 3
3550 C=1
3560 FOR I=1 TO 16
3570 A=INT(RND*16)+1
3580 CALL COLOR(13,C,1)
3590 CALL COLOR(14,3,A)
3600 C=C+1
3610 A=A+1
3620 FOR P=1 TO 8
3630 NEXT P
3640 NEXT I
3650 NEXT R
3660 RETURN
3670 REM AFFICHAGE DE LA TABLE
 ET BONNE REPONSE
3680 CALL CLEAR
3690 FOR B=1 TO 10
3700 PRINT A;"X";B;"=";A*B: :
3710 NEXT B
3720 A$="COURAGE..."
3730 FOR I=1 TO LEN(A$)
3740 CALL HCHAR(12,16+I,ASC(SE
G$(A$,I,1)))
3750 NEXT I
3760 FOR I=1 TO LEN(P$)
3770 CALL HCHAR(14,21+I,ASC(SE
G$(P$,I,1)))
3780 NEXT I
3790 RETURN
3800 PRINT : :"ERREUR !!..";P$
: :"          "&STR$(A)&" fois
 "&STR$(N(R))&" egal
";A*N(R): :
3810 Z=0
3820 GOTO 880
3830 CALL CLEAR
3840 FOR B=11 TO 20
3850 PRINT B;"-";A;"=";B-A: :
3860 NEXT B
3870 A$="ENCORE UN EFFORT."
3880 FOR I=1 TO LEN(A$)
3890 CALL HCHAR(12,14+I,ASC(SE
G$(A$,I,1)))
3900 NEXT I
3910 FOR I=1 TO LEN(P$)
3920 CALL HCHAR(14,18+I,ASC(SE
G$(P$,I,1)))
3930 NEXT I
3940 RETURN
3950 PRINT : :"FAUX!!...";P$:
:"          "&STR$(N(R)+10)&"
moins "&STR$(A)&" eg
al";N(R)+10-A: :
3960 Z=0
3970 GOTO 1490
3980 CALL CLEAR
3990 FOR B=1 TO 10
4000 PRINT A;"+";B;"=";A+B: :
4010 NEXT B
4020 A$="  REVISE!!! "
4030 FOR I=1 TO LEN(P$)
4040 CALL HCHAR(12,16+I,ASC(SE
G$(P$,I,1)))
4050 NEXT I
4060 FOR I=1 TO LEN(A$)
4070 CALL HCHAR(14,20+I,ASC(SE
G$(A$,I,1)))
4080 NEXT I
4090 RETURN
4100 PRINT : :"NON !!...";P$:
:"          "&STR$(A)&" plus "
&STR$(N(R))&" egal";
A+N(R): :
4110 Z=0
4120 GOTO 2100
4130 CALL CLEAR
4140 FOR B=1 TO 10
4150 PRINT B*A;"/";A;"=";B*A/A
: :
4160 NEXT B
4170 A$="ETUDIE!!!"
4180 FOR I=1 TO LEN(A$)
4190 CALL HCHAR(12,18+I,ASC(SE
G$(A$,I,1)))
4200 NEXT I
4210 FOR I=1 TO LEN(P$)
4220 CALL HCHAR(14,20+I,ASC(SE
G$(P$,I,1)))
4230 NEXT I
4240 RETURN
4250 PRINT : :"     NON !!!..."
;P$;: :"     "&STR$(N(R)*A)&"
que divise "&STR$(A)
&" egal";N(R)*A/A: :
4260 Z=0
4270 GOTO 2710
4280 REM FIN DU PROGRAMME
4290 CALL CLEAR
4300 CALL CHAR(128,"3078793276
FCFCFC")
4310 CALL CHAR(129,"307878307F
FCFCFC")
4320 CALL CHAR(130,"FCFCFC4848
484848")
4330 CALL COLOR(13,16,1)
4340 PRINT "          * AU REVOI
R *": : : : : : : : : : : :
4350 FOR I=1 TO 15
4360 CALL SOUND(5,1600,2)
4370 CALL HCHAR(18,13,128,9)
4380 CALL HCHAR(19,13,130,9)
4390 FOR J=1 TO 50
4400 NEXT J
4410 CALL HCHAR(18,13,129,9)
4420 NEXT I
4430 RETURN
```


Les cassettes " 99 Magazine "

1 à 8 contiennent tous les programmes publiés dans les revues correspondantes, à l'exception des sources Assembleur ou Pascal...

Un "bug" dans "GALAXIA"

Une erreur apparaissait sur le listing publié dans le prédédent numéro de "99 Magazine", ainsi que sur le code source placé sur la disquette correspondante. Toutefois, le code objet, baptisé "GALAXIA", était correct. Pour que tout rentre dans l'ordre, il faut modifier, dans le fichier source "GALAXIA-0", la ligne placée juste avant l'étiquette KEY :

BL @VSBW au lieu de BLWP @VSBW

dans le fichier "GALAXIA-6", il faut supprimer la dernière valeur de DATA (,288) dans l'avant dernière ligne.

Quelques adresses

Jean-Claude Hardy, un de nos fidèles lecteurs de Montréal, nous transmet quelques adresses de distributeurs de logiciels pour le TI-99/4A.

- TEX_COMP, P.O. BOX 33084, Granada Hill, Ca. 91344, U.S.A. Tél. : (818) 366-6631.
- TRITON, Products Company, Order Center, P.O. BOX 8123, San Francisco, Ca, 94128. Tél. : (800) 227-6900.
- Compucentre, 9001 Boul. Louis-H. Lafontaine, Montréal, Canada, Tél. : (514) 354-3810.

Petites annonces (gratuites)

• A vendre, TI-99/4A T.B.E. 1000 FF. Module "Basic Etendu" + manuel 500 FF. Extension de mémoire externe 32Ko (01/85) 1100 FF. Magnéto Philips D6600 + câble 550 FF. Paire de joysticks 150 FF. Nombreux livres et cassettes (jeux, utilitaires...). Le tout pour 3000 FF (avec livres et cassettes). Bertrand Delestrez - 4, rue Grissom - 62210 AVION - (21) 28.21.75.

• Vends modules "Parsec", "Burger Time", "Startrek", "Fathom", "Microsurgeon" : 180 FF. "Tombstone", "Video 1", "Wumpus" : 100 FF. Cassettes "Lunar Lander", "Jumper", "Solar System" : 80 FF. Livres. J. Reibel - 9, square Fleming - 92350 LE PLESSIS-ROBINSON - (1) 631.46.11.

• Cherche (échange) programmes pour TI (Basic TI ou Basic Etendu). Réponse assurée. Cherche aussi un manche de manette de jeu. Laurent Kraft - 22, rue Pasteur - 68300 SAINT-LOUIS - (89) 67.51.18.

• Vends TI-99, interface SECAM, manettes de jeux, cordon magnéto, Basic Etendu, Mini-mémoire, manuels, casettes "Basic par soi-même", "Conseils financiers", "Sungames" et 50 programmes sur cassettes. 10 Hebdogiciels. 5 modules de jeux ("Pole position", "Moon Patrol", "Foot", "Carwars", "TI Invaders"). Valeur de l'ensemble 4800 FF, cédé à 3500 FF. Xavier JACQ - 8, rue de la gare - 29223 SAINT THEGONNEC - (98) 79.42.43.

• Achète "Editeur/Assembleur", "Synthétiseur de paroles" et "Terminal Emulator II" pour TI 99/4A. Claude Besneux - BP 56 - 92110 CLICHY - 564.17.74 de 14H à 17H.

• Recherche cassette "Assembleur ligne par ligne" (ou copie) utilisable avec la "Mini-mémoire" (écrasée involontairement et irrécupérable !). Dédommagement assuré. (6) 448.61.81 après 20H.

• Vends module "Gestion de fichiers" (250 FF) et cassette "Conseil financier" (50 FF). Jean-François Dupont - 8, rue Madame de Sévigné - 53000 LAVAL.

• Vends TI-99/4A avec 9 modules : "Basic Etendu", "Echecs", "Othello", "Parsec", "MunchMan", "TI Invaders", "Chasse au Wumpus", "Cars War", "Pirat's Adventure" et synthétiseur de paroles, cordon magnéto, 5 manuels, manettes de jeux. Très bon état. Prix à débattre. Patrick - 657.22.66 entre 18 et 19H.

• Vends boîtier d'extension, lecteur de disquettes et carte contrôleur. Mini-mémoire, livre Assembleur. Cassettes Basic et Basic Etendu par soi-même. Cordon magnéto, 4 modules de jeux, cassettes "Lunar Lander II". 3 livres de programmes. Marc Mathonnet - 33, avenue Auguste Blanqui - 69100 VILLEURBANNE - 16 (7) 885.73.78.

• Vends TI-99/4A, manettes de jeux, magnétophone à cassettes. Guy Forestier - 33910 TAURIAC - (57) 68.86.05 après 18H.

• Vends "Mini-mémoire" et livre "Initiation à l'assembleur" : 750 FF. module "Foot" : 250 FF, "Music-Maker" : 250 FF. François Frattini - 30, galerie de l'Arlequin -Appt. 2224 - 38100 GRENOBLE - (76) 22.16.12.

• Vends TI-99/4A, "Basic Etendu" en français, livres et cassettes : 2000 FF. (70) 20.13.84 ou (37) 97.21.72 le week end.

• Cherche modules "Mini-mémoire" avec cassette et "Basic Etendu". Paul Ducros - 55, rue des Loriots - 03100 MONTLUCON.

• Cherche contacts sur SAUMUR et sa région pour tester programme. (41) 38.11.90 le matin.

• Vends "Mini-mémoire" avec manuel "Editeur/Assembleur" et "Basic Etendu". (1) 704.43.17.

• Achète module "Tunnels of Doom" + 2 cassettes (environ 250 FF). Achète ou échange cassettes "Mistery Fun House", "Mission Impossible", "Voodoo Castle" (je peux échanger contre d'autres jeux d'Adventure International"). Laurent Pelé - 5, clos d'Ergal - 78310 ELANCOURT - (3) 051.18.96.

Courrier des Lecteurs

Alexandre Dubach

Je vous écris pour vous féliciter et ensuite vous poser quelques questions :

Comment, avec l'assembleur (mini-mémoire et extension de mémoire) envoyer quelque chose vers l'imprimante ? existe-t-il une adresse ? Je possède personnellement l'imprimante Seikosha GP100A. Comment alors simuler le CHR$(8) (accès au mode graphique) ?

Ma seconde question porte sur la routine DSRLNK (à l'adresse >6038 pour la mini-mémoire) : à quoi sert-elle ? Je crois qu'elle n'est même pas mentionnée dans le livre de Denise Amrouche et Roger Didi intitulé "Initiation au langage assembleur de l'ordinateur familial Texas Instruments".

J'aimerais savoir si, en mode 40 colonnes (par l'assembleur), on peut, simplement et sans revenir systématiquement au mode 32 colonnes, mettre en mémoire (ou afficher sur l'écran) une lettre frappée sur le clavier.

Stéphane Dunglas - 121, rue de la Pompe - 75116 Paris.

Votre seconde question constitue une réponse à la première. En effet, la routine DSRLNK est utilisée pour effectuer toutes les sorties (ou entrées) vers (ou depuis) les périphériques. Nous avions prévu un article consacré aux diverses utilisations de cette routine mais, faute de place, il nous était impossible de le publier dans ce numéro; vous le trouverez donc dans le numéro 9. En attendant, des exemples concrets sont à votre disposition dans les programmes "Désassembleur" (numéro 5) et "Routines graphiques" (numéro 7).

Le clavier et le processeur vidéo sont en fait deux "périphériques" totalement indépendants; il est donc possible d'obtenir un affichage 40 colonnes tout en ayant toujours accès au clavier. Pour vous en convaincre, vous pouvez consulter le programme "Désassembleur" (encore !) du numéro 5.

Je vous communique une petite adresse utile :

En Basic Etendu (avec extension de mémoire) ou mini-mémoire, entrer un programme (n'importe quoi), puis :

CALL INIT
CALL PEEK(-31952,A,B,C,D)
PRINT A,B,C,D
(notez les valeurs affichées)
NEW (plus de programme !)

Tapez maintenant :

CALL LOAD(-31952,A,B,C,D)
(A, B, C et D étant les valeurs trouvées précédemment) et
LIST
Votre programme est toujours là.

Conclusion : le NEW n'efface pas le programme en mémoire (cela marche même si vous éteigner la console, mais pas le boîtier d'extension).

Omar Belmoktar - 6, allée S. de Brosse - 93150 Le Blanc Mesnil.

Est-il possible de connecter une table traçante sur l'interface RS232.

Docteur G. Lourde - 8, place du Commando - 44600 St Nazaire Cidex.

L'interface RS232 du TI-99 est standard et, de plus, configurable par logiciel; une table traçante répondant à ce standard doit donc fonctionner normalement sur le TI. Cependant, si vous projetez l'acquisition d'un tel appareil, nous ne saurions trop vous conseiller de faire un essai chez votre revendeur.

Pourriez-vous m'indiquez les noms des MODEMs pouvant être raccordés au TI-99/4A.

Arnaud Duminy - chemin des Ecus - 59600 Maubeuge.

On peut, par l'intermédiaire de l'interface série, raccorder n'importe quel MODEM au TI. Il existe de nombreux MODEM à des prix et avec des possibilités diverses; le mieux serait donc de consulter votre revendeur qui, en fonction de votre problème, pourrait vous conseiller un modèle plutôt qu'un autre.

Je ne possède qu'un seul lecteur de disquette. Quelle est la marche à suivre pour copier une disquette sur une autre disquette vierge. Par exemple, un fichier utilisé pour un annuaire téléphonique, ou encore la disquette "Editeur / Assembleur" pour laquelle le manuel conseille une copie avant toute utilisation.

Jean Langlet - 18, rue E. Brasselet - Leval - 59620 Aulnoye Aymeries.

Le module "Gestion de disques" est prévu pour ce type d'utilisation. Dans le menu principal (après le menu proposant le choix de la langue utilisée pour les messages), il faut choisir l'option "INITIALISATION POUR UTILISATION AVEC UN SEUL DISQUE". Après cela, il vous suffit de choisir l'option "COPIE DE DISQUE"; le système affichera à l'écran toutes les indications nécessaires.

Etant possesseur d'un TI-99, j'aimerai savoir si un lecteur de disquettes de marque QUMETRACH ou SHUGGART qui utilisent des disquettes double face et double densité peuvent être installés sur le Texas.

Jean-Claude Siracis - 11, rue de la République - 11180 Ouvrillan.

Si vous disposez du contrôleur de disquettes Texas Instruments, le raccordement de ces lecteurs ne devrait pas présenter de difficulté.

Lecteur assidu de "99 Magazine", j'ai lu dans le numéro 7 qu'il était possible de brancher une imprimante ImageWriter sur un TI-99/4A.
Je dispose de la Mini-mémoire, du Basic Etendu, de l'extension 32Ko et de la carte RS232C.
L'imprimante Apple s'adapte-t-elle directement sur le port série de la carte ? Si oui, comment faut-il programmer la carte pour obtenir un listing de programmes écrits en Basic ou Basic Etendu ?

Georges Chareyron - 4, rue F. Buisson - 07400 Le Teil.

L'imprimante Apple "ImageWriter" est aussi facile à utiliser qu'une Epson ou Seikosha. Un câble standard RS232 suffit à son raccordement. Pour obtenir un listing à partir du Basic :

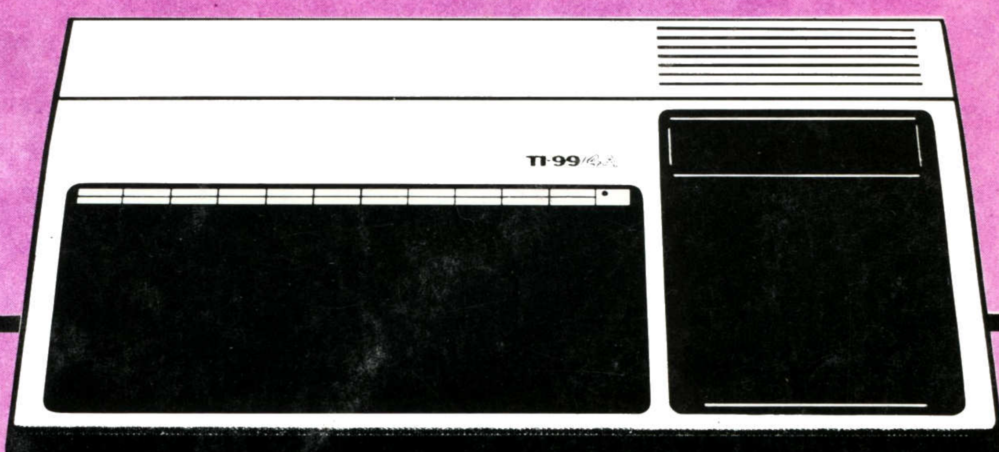
LIST "RS232.BA=9600" (9600 bauds est la valeur par défaut pour l'Image-Writer).

□

CASSETTE
D'ACCOMPAGNEMENT
DISPONIBLE

99 MAGAZINE

Gestion de fichiers
transfert image/cassette
Musique et Mini-mémoire
Pointeurs et variables en Pascal



Numéro 9 - Trimestriel - Juin 1985 - 40 F

ISSN 0759-9005

Sommaires des numéros 1 à 8 de "99 Magazine"

Numéro 1

Musik Maker à l'essai
Pourquoi acheter un TI-99/4A
A vous de programmer
Déballage du TI-99
Les avantages du Basic étendu — BE
Initiation au Basic — BTI
Jeu du Pendu — BTI
Fabriquez votre joystick
Appelez moi KEY — BTI/BE
Programmes d'aide à la décision — BE
Jeu de l'Alphabet — BTI
Le club d'utilisateurs TI-Education
Introduction au LOGO — L
Les tours de Hanoï — BTI
Récupérez les programmes Apple — BTI/BE
Le troisième menuet de Bach — BTI
Le dernier robot — BTI
Cherchez l'ancêtre — BTI
Gestion personnelle de fichiers
Quand le TI devient un harmonium — BTI
Fichiers en Basic — BE

Numéro 2

Micro-monde LOGO : les lutins — L
La bataille navale — BTI
Le crocodile savant
Jeu du Simon — BTI/BE
Jeu des carrés — BTI
L'ordinateur de l'avenir
TI-Writer à l'essai
Canon de Purcell — BE
Chicken Helper — BE
Sauvegarde des fichiers et prog. — BTI/BE
Le Mastermind — BTI
Othello — BTI
Astéroïdes — BE
Kermit la grenouille — BE
Le Biorythme — BTI

Numéro 3

Un programme de calcul — BE
Annonce de Texas Instruments
Othello à l'essai
Apprendre Basic en programmant (1) — BTI
Introduction au Pascal — P
Puissance 4 — BE
Les modes d'adressage en Assembleur — A
Géographie : la carte de France — BTI
Créez votre touche RESET
Puzzle — BTI
Awari — BTI
Le sapin de Noël — BE
PARSEC à l'essai
LOGO, seulement pour enfant ? — L
Le Mastermind en assembleur — A
Du nouveau sur les cassettes
Le magnétophone TI

Numéro 4

SOS Hélicoptère — BE
Utilisations de la Mini-mémoire — BTI
Communiqué de Texas Instruments
Le serveur de restaurant — BTI
"Echecs" à l'essai
Combat de chars — BTI
Réalisez votre cordon de magnéto
Apprendre Basic en programmant (2) — BTI
Jeu de mémorisation — BTI
Jack-pot — BE
Les nombres en LOGO — L
Graphisme en haute résolution — BTI + A
Sous-programmes en assembleur — A
Horloge — BE
Un détecteur de lumière pour le TI
Le Club M.T.I.
Le système P-UCSD — P
Les lutins du Basic étendu — BE
Le solitaire — BTI
Claustrophobia — BTI
"DISPLAY" et "ACCEPT" avec PRK — BTI
Une erreur dans le manuel MINIMEM
Réparez votre assembleur ligne par ligne — A

Numéro 5

Conversions — BTI
Systèmes linéaires — BTI
Structure des disques du P-Système — P
Poèmes et graphismes — BTI
Un désassembleur en assembleur — A
Les tours de Hanoï — BE
Trouvez le bon mot — BTI
Le jeu de la vie — BTI ou A
Programmathèque
Les opérateurs logiques — BE
Apprendre Basic en programmant (3) — BTI
Bibliographie
Warangal — BTI
Graphisme en haute résolution — BTI + A
Le trésor de l'Armada — BE
Ecriture littérale d'un nombre — P
Copie d'écran sur imprimante Epson — BTI
Description du port d'entrées/sorties
Translet — BE
Le Club Ordinabis
Le Club Ticazur

Numéro 6

Des chiffres et des lettres — BTI
A l'heure du Pascal — P
Kong — BE
Mini-mémoire moins "mini"
Edipage — BTI
Caractères ASCII/page graphique — BTI + A
Musik — BTI
Copie d'écran sur Seïkosha GP-100 — BE
Nouvelles fonctions avec MMEM — BTI + A
Le Basic à la loupe — BTI ou BE
Star Treck — BE
Ecriture directe dans registres VDP — BTI
My tailor was rich — BTI
Le code de la route — BE
Un double face sur votre TI

Numéro 7

La défausse — BTI
Le tableur TI-Calc
Sauvegarde d'images graphiques — A
Copie d'écran — A
Copie d'écran sur Seïkosha GP-250 — BE
La compilation séparée — P
Les bibliothèques — P
Trouvez votre chemin dans les jeux d'aventure TI — A
Srand — BTI
Galaxia — A
LOGO et la récursivité — L
Dessinez avec le Basic Etendu — BE
Accès au mode BitMap en Basic TI — BTI
Music Box Dancer — BTI
Des nouvelles du Club Ticazur
Routines graphiques pour le BE — BE + A

Numéro 8

Bit-Map et Mini-mémoire — BTI + A
Le train sifflera trois fois... — BE
Comptes familiaux — BE
Mission spéciale — BE
Crayon optique — A
Le traitement des fichiers en Pascal — P
Editeur de caractères — BE
Kaleïdoscope — BE
Opérations — BTI

* **BTI** : Basic TI. **BE** : Basic étendu. **A** : Assembleur. **P** : Pascal. **L** : Logo.

Tous les numéros sont disponibles.

99 Magazine numéro 9

Starline — Georges Goument — 25

Editorial — Hervé Thiriez — 5

Pointeurs et variables dynamiques en Pascal — Gérard Santraille — 11

Transfert image/cassette — Gérard Baroni-Jean Marin — 22

Ticrok — Georges Goument — 58

Courrier des Lecteurs — Alexandre Duback — 65

Gestion de fichiers en Basic étendu et assembleur — Michel Teyssou — 31

Un nouveau module graphique — Henri Mathian — 58

Musique et Mini-mémoire — Denise Amrouche — 19

Petites annonces — 66

Module Maximem — 64

Pentaxe — Georges Goument — 6

Editions MEV - 64/70, rue des Chantiers - 78000 Versailles Tél. : (3) 951.24.43 Directeur de la publication : Hervé Thiriez

Editorial

Ça ne va pas très bien, en ce moment, dans le petit monde de la micro-informatique. Pratiquement tous les constructeurs d'ordinateurs - familiaux ou non - licencient une partie, souvant loin d'être négligeable, de leur personnel. Même IBM, qui n'est pas la plus petite société dans ce domaine, n'échappe pas à cette vague de morosité. La presse spécialisée souffre de cet état de fait, puisque les contrats de publicité se font plus rares, et il faut savoir que les revenus de pratiquement toutes les revues proviennent, pour la plus grande part, de la pub. Dans ce contexte, il est problable que quelques publications spécialisées en micro-informatique seront contraintes à cesser leurs activités.

Heureusement, la politique commerciale et éditoriale des Editions MEV fait que "99 Magazine" n'a pas besoin de publicité pour vivre ; donc, ça va plutôt bien pour nous, même si ce numéro paraît très en retard, cela à cause de problèmes techniques et de personnel. Nous vous prions de bien vouloir nous excuser pour ce décalage, mais je pense que vous serez d'accord avec moi, il vaut mieux un "99" en retard que pas de "99" du tout. Le fait que "99 Magazine" marche bien nous pose quelques problèmes d'intendance ; en effet, nous nous sommes trouvés en rupture de stock pour quelques disquettes et cassettes, en particulier la 6, mais nous espérons que tout sera rentré dans l'ordre lorsque vous lirez ces lignes.

*Après ces bonnes paroles, voyons un peu ce que contient ce numéro. Certains d'entre-vous nous ont reproché de ne pas avoir assez mis de programmes en Basic TI dans le numéro 8 ; vous trouverez donc cette fois trois programmes écrits en ce langage ("Starline", "Ticrok" et "Pentaxe") par **Georges Goument**. Vous saurez beaucoup de choses sur les transferts d'informations entre le TI-99/4A et un magnétophone, après avoir lu l'article de **Gérard Baroni** et **Jean Marin**, les deux "piliers" du Club Ticazur, dont il n'est pas inutile de rappeler les coordonnées : Club Ticazur - Gérard Baroni - Villa Héléna - Chemin des Bas-Campons - 06480 La colle-sur-Loup - Tél. : 32.63.16.*

***Gérard Santraille** vous communique un maximum d'informations sur les pointeurs et variables dynamiques en Pascal, alors que **Denise Amrouche** vous indique un moyen simple et efficace pour faire cohabiter musique et programmes Basic. **Henri Mathian** vous présente le module "Apesoft", qui est un Basic étendu à vocation graphique distribué en France par "la règle à calcul", qui distribue aussi depuis peu des lecteurs et contrôleurs de disquettes pour TI-99. "99 Magazine" vous présente aussi un tout nouveau module - il n'est pas encore commercialisé - le module "Maximem 48Ko".*

*Outre les rubriques habituelles "Courrier des Lecteurs", par **Alexandre Duback**, et "Petites annonces" vous trouverez deux programmes de gestion de fichiers écrits par **Michel Teyssou**, l'un écrit en assembleur, l'autre en Basic étendu. La juxtaposition de deux programmes identiques, mais écrits en deux langages diamétralement opposés présente un intérêt pédagogique indéniable.*

"99 Magazine" sera à la boutique Sicob (Porte Maillot - Stand M279), nous vous-y attendrons !

Hervé Thiriez

Rédacteur en chef - Directeur de la publication : Hervé Thiriez. **Rédaction** : Jean-Luc Bazanegue, Gérard Santraile. **Dessins** : Laurent Bidot. **Ont collaboré à ce numéro** : Denise Amrouche, Gérard Baroni, Jean-Luc Bazanegue, Alexandre Duback, Georges Goument, Jean Marin, Henri Mathian, Gérard Santraille, Michel Teyssou.

Editions MEV - 64/70, rue des Chantiers - 78000 Versailles - Tél. : (3) 951.24.43.

Publicité : consultez les Editions MEV.

Impression : Imprimerie du Lion - Repro-Versailles - 25, rue Michel Lecomte - 75003 Paris - Tél. : 272.96.19.

Pentaxe

Georges Goument

Ce jeu de tactique est adapté du morpion Coréen. Nous vous proposons quelques commentaires, afin de compléter les règles du jeu affichées par le programme.

- Il est possible de prendre plusieurs paires de pions adverses en une seule fois. A vous de jouer afin d'obtenir cette possibilité.
- Après une prise, l'adversaire peut rejouer aux mêmes emplacements sans risques, contrant ainsi le jeu de celui qui vient de prendre.
- Pour réaliser un alignement, l'emplacement du dernier pion posé n'a aucune importance ; le programme prend en compte les cinq positions possibles.

Ne vous emmêlez pas les touches en tapant les lignes 339 à 407 ; le déroulement du jeu serait probablement compromis.

Il ne vous reste qu'à trouver un adversaire de valeur pour passer quelques bons moments.

Déroulement du programme

27 à 80 : règles du jeu.
81 à 104 : redéfinition des caractères.
105 à 108 : variables P1/P2 = prises pions adverses ; G1/G2 = parties gagnées.
109 à 151 : dessin du jeu.
152 à 173 : changement de joueur. Variables A et B = pions selon joueur, change la couleur d'affichage selon J=0 ou J=1.
174 à 176 : affiche la première coordonnée (ligne=Y) côté gauche ou droit.
177 à 284 : attente d'une touche ; A à Q = coordonnées du pion à jouer. La touche zéro annule la coordonnée Y. La touche <ENTER> annule la partie.
285 - 286 : affiche la seconde coordonnée (colonne=X).
287 à 300 : vérifie que la position jouée est libre. Sert également pour la touche "0" (AN=1).
301 à 305 : le pion est posé sur le jeu ; permute J pour le tour suivant.
306 à 338 : contrôle dans les huit directions possibles, la présence de pions sur quatre intersections maximum.
339 à 407 : selon positions et couleurs des pions détectés, détermine :

- soit une prise (avec retour pour suite du programme) ;
- soit un alignement (partie gagnée, envoi ligne 422) ;
- soit un renvoi ligne 152 pour continuer la partie.

408 à 421 : prise d'une paire de pions. Affichage selon le joueur à droite ou à gauche face à P.
422 à 465 : alignement réalisé ; texte vert ou rouge selon le vainqueur. Affichage identique prise face à G.
466 à 471 : annulation d'une partie.
472 à 478 : nouvelle partie ou arrêt du jeu.
479 à 487 : résultat final (10 parties maximum).
488 à 500 : tirage au sort du premier joueur. Ensuite, donne le jeu au perdant.

99

Programme Pentaxe

Basic TI

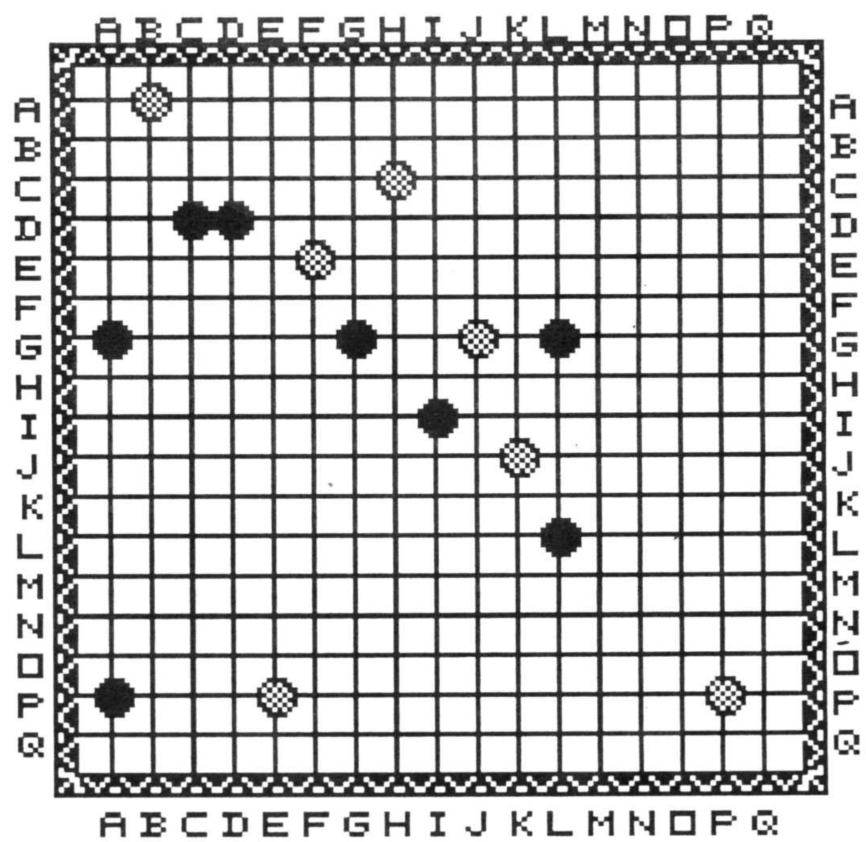
```
10 REM ***************
11 REM *             *
12 REM *   PENTAXE   *
13 REM *             *
14 REM *  programme  *
15 REM *     en      *
16 REM *  Basic   TI *
17 REM *             *
18 REM *  copyright  *
19 REM *             *
20 REM * 99 Magazine *
21 REM *             *
22 REM *      &      *
23 REM *             *
24 REM * G.  Goument *
25 REM *             *
26 REM ***************
27 CALL CLEAR
28 FOR I=1 TO 12
29 CALL COLOR(I,1,1)
30 NEXT I
31 CALL CHAR(33,"FFFFF
   FFFFFFFFFFF")
32 PRINT TAB(9);"*****
   ******":TAB(9) ;"*!
   !!!!!!!!*":TAB(9);"
   *!PENTAXE!*":TAB(9)
   ;"*!!!!!!!!!*"
33 PRINT TAB(9);"*****
   ******"::TAB(11);"j
   eu de"::TAB(10);"ta
   ctique"::TAB(9) ;"2
      joueurs"::::
34 PRINT "<c> g goumen
   t decembre 1984"::::
35 CALL COLOR(1,12,1)
36 CALL SCREEN(8)
37 CALL COLOR(2,10,14)
38 FOR I=3 TO 12
39 CALL COLOR(I,13,1)
40 NEXT I
41 FOR I=5 TO 8
42 CALL COLOR(I,7,12)
43 NEXT I
44 RESTORE 48
45 FOR I=1 TO 53
46 READ D,N
```

```
47 CALL SOUND(-50*D,N*
   6,8,N*4,8,N/2 ,5)
48 DATA 2,523,2,587,2,
   587,2,587,2, 587,2,
   523,4,622,2,587,2,5
   23,2,659,2,784,2,10
   47,4,1047,2,1047,2,
   880
49 DATA 2,784,2,698,2,
   1047,2,698,2,659,2,
   659,2,587,2,698,2,6
   59,4,587,2,587,2,52
   3,2,587,2,659,2,784
50 DATA 2,784,2,659,4,
   784,2,784,2, 698,2,
   440,2,440,2,349,3,4
   40,2,440,2,262,2,29
   4,2,330,2,392,2,330
51 DATA 4,392,2,392,2,
   349,2,349,2, 220,2,
   220,4,349,6,220
52 NEXT I
53 CALL CLEAR
54 FOR I=5 TO 8
55 CALL COLOR(I,6,1)
56 NEXT I
57 CALL COLOR(2,13,1)
58 PRINT "    PRESENTA
   TION DU JEU":"    ~
   ~~~~~~~~~~~~~~~~~~"
   :::"plateau de jeu
   forme de 289 inters
   ections."::
59 PRINT "  chacun des
    joueurs disposede
   pions verts ou roug
   es."::"  le jeu con
   siste a placer 5"
60 PRINT "de ces pions
    sur une ligne."::"
      cet alignement
   peut etrehorizontal
   , vertical ou bien"
61 PRINT "en diagonale
   ."::"avec la prise
   de 5 paires depions
    adverses la partie
    estegalement gagne
   e...."::::
62 GOSUB 72
63 PRINT "choisissez v
   otre couleur au deb
   ut de la partie."::
   "l'ordinateur desig
   nera celle"
64 PRINT "qui doit com
   mencer en posantle
   premier pion au cen
   tre duplateau de je
   u."::
65 PRINT "  pour toute
    nouvelle partiela
   couleur perdante jo
   ue en premier."::::
66 GOSUB 72
67 PRINT "    PLACEMEN
   T DES PIONS":"
   ~~~~~~~~~~~~~~~~~~~
   ":::"tapez deux let
   tres face aux"
68 PRINT "intersection
   s desirees."::"1ere
    lettre --> LIGNE":
   :"2eme lettre --> C
   OLONNE"::
69 PRINT "touche ENTER
    pour annuler lapar
   tie."::"un appui su
   r ZERO annule la co
   ordonnee de la lign
   e."::::
70 GOSUB 72
71 GOTO 81
72 CALL HCHAR(24,28,45
   ,2)
73 CALL HCHAR(24,30,62
   )
74 CALL HCHAR(24,26,83
   )
75 CALL KEY(3,T,S)
76 CALL HCHAR(24,26,32
   )
77 IF S=0 THEN 74
78 IF T<>83 THEN 75
79 CALL CLEAR
80 RETURN
81 DIM H(32)
82 RANDOMIZE
83 RE=1
84 CALL CLEAR
85 M$="Veuillez patien
   ter   S.V.P"
86 GOSUB 457
87 CALL HCHAR(24,3,32,
   28)
88 CALL SCREEN(2)
89 CALL CHAR(64,"00103
   07F3010")
90 CALL CHAR(65,"00080
   CFE0C08")
91 CALL CHAR(66,"00103
   87C1010101")
92 FOR I=72 TO 80
93 READ A$
94 CALL CHAR(I,A$)
95 NEXT I
96 DATA FFB3D6E58FB8E8
   D8,FF66BDDBFF1 0101
   0,FFCD6BA7F11D171B,
   B8D8E8BFB0E8D8B8,10
   1010FF10101010
97 DATA 1D1B17FD0D171B
   1D,D8E8B88FE5D 6B3F
   F,101010FFDBBD66FF,
   1B171DF1A76BCDFF
98 CALL CHAR(40,"3C7EF
   FFFFFFF7E3C")
99 CALL CHAR(41,"FFFFF
   FFFFFFFFFFF")
100 CALL CHAR(88,"3C7E
    FFFFFFFF7E3C")
101 CALL CHAR(89,"FFFF
    FFFFFFFFFFFF")
102 FOR I=1 TO 12
103 CALL COLOR(I,1,1)
104 NEXT I
105 P1=48
106 P2=48
107 G1=48
108 G2=48
109 A=97
110 FOR I=5 TO 21
111 CALL HCHAR(I,6,A)
112 CALL HCHAR(I,26,A)
113 A=A+1
114 NEXT I
115 A=97
116 FOR I=8 TO 24
117 CALL HCHAR(3,I,A)
118 CALL HCHAR(23,I,A)
119 A=A+1
120 NEXT I
121 CALL HCHAR(4,7,72)
122 CALL HCHAR(4,8,73,
    17)
123 CALL HCHAR(4,25,74
    )
124 CALL HCHAR(22,7,78
    )
125 CALL HCHAR(22,8,79
    ,17)
126 CALL HCHAR(22,25,8
    0)
127 CALL VCHAR(5,7,75,
    17)
128 CALL VCHAR(5,25,77
    ,17)
129 Q=76
130 CALL HCHAR(4,3,112
    )
131 CALL HCHAR(4,29,11
    2)
132 CALL HCHAR(8,3,103
    )
133 CALL HCHAR(8,29,10
    3)
134 CALL HCHAR(6,3,P1)
135 CALL HCHAR(6,29,P2
    )
136 CALL HCHAR(10,3,G1
    )
137 CALL HCHAR(10,29,G
    2)
138 FOR I=8 TO 24
```

```
139 CALL VCHAR(5,I,Q,1
    7)
140 NEXT I
141 CALL COLOR(1,2,2)
142 CALL COLOR(2,13,15
    )
143 CALL COLOR(3,8,1)
144 CALL COLOR(4,8,1)
145 CALL COLOR(6,5,15)
146 CALL COLOR(7,5,15)
147 CALL COLOR(8,9,15)
148 FOR I=9 TO 11
149 CALL COLOR(I,12,1)
150 NEXT I
151 GOTO 488
152 IF J=1 THEN 161
153 M=65
154 Z=5
155 A=40
156 B=88
157 CO=3
158 CALL HCHAR(2,28,32
    ,2)
159 CALL HCHAR(2,3,41,
    2)
160 GOTO 168
161 M=64
162 Z=27
163 A=88
164 B=40
165 CO=10
166 CALL HCHAR(2,3,32,
    2)
167 CALL HCHAR(2,28,89
    ,2)
168 CALL COLOR(5,CO,16
    )
169 CALL SOUND(3,1966,
    11)
170 CALL SOUND(10,1949
    ,11)
171 Y=0
172 X=0
173 GOTO 177
174 CALL HCHAR(Y,Z,M)
175 CALL SOUND(1,770,1
    0)
176 C=1
177 CALL KEY(3,T,S)
178 IF S=0 THEN 177
179 IF T<>48 THEN 182
180 AN=1
181 GOTO 288
182 IF T<>13 THEN 184
183 GOTO 466
184 IF T<>65 THEN 190
185 IF C=1 THEN 188
186 Y=5
187 GOTO 174
188 X=8
189 GOTO 285
190 IF T<>66 THEN 196
191 IF C=1 THEN 194
192 Y=6
193 GOTO 174
194 X=9
195 GOTO 285
196 IF T<>67 THEN 202
197 IF C=1 THEN 200
198 Y=7
199 GOTO 174
200 X=10
201 GOTO 285
202 IF T<>68 THEN 208
203 IF C=1 THEN 206
204 Y=8
205 GOTO 174
206 X=11
207 GOTO 285
208 IF T<>69 THEN 214
209 IF C=1 THEN 212
210 Y=9
211 GOTO 174
212 X=12
213 GOTO 285
214 IF T<>70 THEN 220
215 IF C=1 THEN 218
216 Y=10
217 GOTO 174
218 X=13
219 GOTO 285
220 IF T<>71 THEN 226
221 IF C=1 THEN 224
222 Y=11
223 GOTO 174
224 X=14
225 GOTO 285
226 IF T<>72 THEN 232
227 IF C=1 THEN 230
228 Y=12
229 GOTO 174
230 X=15
231 GOTO 285
232 IF T<>73 THEN 238
233 IF C=1 THEN 236
234 Y=13
235 GOTO 174
236 X=16
237 GOTO 285
238 IF T<>74 THEN 244
239 IF C=1 THEN 242
240 Y=14
241 GOTO 174
242 X=17
243 GOTO 285
244 IF T<>75 THEN 250
245 IF C=1 THEN 248
246 Y=15
247 GOTO 174
248 X=18
249 GOTO 285
250 IF T<>76 THEN 256
251 IF C=1 THEN 254
252 Y=16
253 GOTO 174
254 X=19
255 GOTO 285
256 IF T<>77 THEN 262
257 IF C=1 THEN 260
258 Y=17
259 GOTO 174
260 X=20
261 GOTO 285
262 IF T<>78 THEN 268
263 IF C=1 THEN 266
264 Y=18
265 GOTO 174
266 X=21
267 GOTO 285
268 IF T<>79 THEN 274
269 IF C=1 THEN 272
270 Y=19
271 GOTO 174
272 X=22
273 GOTO 285
274 IF T<>80 THEN 280
275 IF C=1 THEN 278
276 Y=20
277 GOTO 174
278 X=23
279 GOTO 285
280 IF T<>81 THEN 177
281 IF C=1 THEN 284
282 Y=21
283 GOTO 174
284 X=24
285 CALL HCHAR(24,X,66
    )
286 C=0
287 CALL SOUND(1,770,1
    0)
288 CALL HCHAR(Y,Z,32)
289 IF AN=1 THEN 295
290 CALL HCHAR(24,X,32
    )
291 CALL GCHAR(Y,X,F)
292 IF F=40 THEN 295
293 IF F=88 THEN 295
294 GOTO 301
295 FOR I=1 TO 5
296 CALL SOUND(10,110,
    0)
297 NEXT I
298 AN=0
299 C=0
300 GOTO 177
301 CALL HCHAR(Y,X,A)
302 IF J=1 THEN 305
303 J=1
304 GOTO 306
305 J=0
306 CALL GCHAR(Y-1,X-1
    ,H1)
307 CALL GCHAR(Y-2,X-2
    ,H2)
308 CALL GCHAR(Y-3,X-3
```

```
    ,H3)
309 CALL GCHAR(Y-4,X-4
    ,H4)
310 CALL GCHAR(Y-1,X,H
    5)
311 CALL GCHAR(Y-2,X,H
    6)
312 CALL GCHAR(Y-3,X,H
    7)
313 CALL GCHAR(Y-4,X,H
    8)
314 CALL GCHAR(Y-1,X+1
    ,H9)
315 CALL GCHAR(Y-2,X+2
    ,H10)
316 CALL GCHAR(Y-3,X+3
    ,H11)
317 CALL GCHAR(Y-4,X+4
    ,H12)
318 CALL GCHAR(Y,X+1,H
    13)
319 CALL GCHAR(Y,X+2,H
    14)
320 CALL GCHAR(Y,X+3,H
    15)
321 CALL GCHAR(Y,X+4,H
    16)
322 IF Y>20 THEN 335
323 CALL GCHAR(Y+1,X+1
    ,H17)
324 CALL GCHAR(Y+2,X+2
    ,H18)
325 CALL GCHAR(Y+3,X+3
    ,H19)
326 CALL GCHAR(Y+4,X+4
    ,H20)
327 CALL GCHAR(Y+1,X,H
    21)
328 CALL GCHAR(Y+2,X,H
    22)
329 CALL GCHAR(Y+3,X,H
    23)
330 CALL GCHAR(Y+4,X,H
    24)
331 CALL GCHAR(Y+1,X-1
    ,H25)
332 CALL GCHAR(Y+2,X-2
    ,H26)
333 CALL GCHAR(Y+3,X-3
    ,H27)
334 CALL GCHAR(Y+4,X-4
    ,H28)
335 CALL GCHAR(Y,X-1,H
    29)
336 CALL GCHAR(Y,X-2,H
    30)
337 CALL GCHAR(Y,X-3,H
    31)
338 CALL GCHAR(Y,X-4,H
    32)
339 IF (H1<>B)+(H2<>B)
    +(H3<>A)THEN 345
340 D=Y-1
341 E=X-1
342 F=Y-2
343 G=X-2
344 GOSUB 408
345 IF (H1=A)*(H2=A)*(
    H3=A)*(H4=A) THEN
    422
346 IF (H3=A)*(H2=A)*(
    H1=A)*(H17=A) THEN
     422
347 IF (H2=A)*(H1=A)*(
    H17=A)*(H18=A) THE
    N 422
348 IF (H1=A)*(H17=A)*
    (H18=A)*(H19= A)TH
    EN 422
349 IF (H5<>B)+(H6<>B)
    +(H7<>A)THEN 355
350 D=Y-1
351 E=X
352 F=Y-2
353 G=X
354 GOSUB 408
355 IF (H5=A)*(H6=A)*(
    H7=A)*(H8=A) THEN
    422
356 IF (H7=A)*(H6=A)*(
    H5=A)*(H21=A) THEN
     422
357 IF (H6=A)*(H5=A)*(
    H21=A)*(H22=A) THE
    N 422
358 IF (H5=A)*(H21=A)*
    (H22=A)*(H23= A)TH
    EN 422
359 IF (H9<>B)+(H10<>B
    )+(H11<>A)THEN 365
360 D=Y-1
361 E=X+1
362 F=Y-2
363 G=X+2
364 GOSUB 408
365 IF (H9=A)*(H10=A)*
    (H11=A)*(H12= A)TH
    EN 422
366 IF (H11=A)*(H10=A)
    *(H9=A)*(H25= A)TH
    EN 422
367 IF (H10=A)*(H9=A)*
    (H25=A)*(H26= A)TH
    EN 422
368 IF (H9=A)*(H25=A)*
    (H26=A)*(H27= A)TH
    EN 422
369 IF (H13<>B)+(H14<>
    B)+(H15<>A) THEN 3
    75
370 D=Y
371 E=X+1
372 F=Y
373 G=X+2
374 GOSUB 408
375 IF (H13=A)*(H14=A)
    *(H15=A)*(H16= A)T
    HEN 422
376 IF (H15=A)*(H14=A)
    *(H13=A)*(H29= A)T
    HEN 422
377 IF (H14=A)*(H13=A)
    *(H29=A)*(H30= A)T
    HEN 422
378 IF (H13=A)*(H29=A)
    *(H30=A)*(H31= A)T
    HEN 422
379 IF (H17<>B)+(H18<>
    B)+(H19<>A) THEN 3
    85
380 D=Y+1
381 E=X+1
382 F=Y+2
383 G=X+2
384 GOSUB 408
385 IF (H17=A)*(H18=A)
    *(H19=A)*(H20 =A)T
    HEN 422
386 IF (H21<>B)+(H22<>
    B)+(H23<>A) THEN 3
    92
387 D=Y+1
388 E=X
389 F=Y+2
390 G=X
391 GOSUB 408
392 IF (H21=A)*(H22=A)
    *(H23=A)*(H24 =A)T
    HEN 422
393 IF (H25<>B)+(H26<>
    B)+(H27<>A) THEN 3
    99
394 D=Y+1
395 E=X-1
396 F=Y+2
397 G=X-2
398 GOSUB 408
399 IF (H25=A)*(H26=A)
    *(H27=A)*(H28 =A)T
    HEN 422
400 IF (H29<>B)+(H30<>
    B)+(H31<>A) THEN 4
    06
401 D=Y
402 E=X-1
403 F=Y
404 G=X-2
405 GOSUB 408
406 IF (H29=A)*(H30=A)
    *(H31=A)*(H32 =A)T
    HEN 422
407 GOTO 152
408 CALL SOUND(300,-4,
    0)
409 CALL HCHAR(F,G,Q)
410 CALL HCHAR(D,E,Q)
```

```
411 IF J=1 THEN 417
412 P2=P2+1
413 CALL SOUND(5,1200,
    0)
414 CALL HCHAR(6,29,P2
    )
415 IF P2=53 THEN 422
416 RETURN
417 P1=P1+1
418 CALL SOUND(5,1200,
    0)
419 CALL HCHAR(6,3,P1)
420 IF P1=53 THEN 422
421 RETURN
422 IF J=1 THEN 428
423 P$=" rouges "
424 CO=9
425 G2=G2+1
426 IF G2>57 THEN 479
427 GOTO 432
428 P$=" verts "
429 CO=13
430 G1=G1+1
431 IF G1>57 THEN 479
432 CALL SCREEN(16)
433 FOR I=9 TO 12
434 CALL COLOR(I,CO,1)
435 NEXT I
436 FOR REP=1 TO 4
437 CALL SCREEN(2)
438 CALL SOUND(-50,262
    ,10,196,15, 167,20
    )
439 CALL SCREEN(5)
440 CALL SOUND(-50,330
    ,10,247,15, 208,20
    )
441 CALL SCREEN(9)
442 CALL SOUND(-50,247
    ,10,185,15, 156,20
    )
443 CALL SCREEN(12)
444 CALL SOUND(-50,220
    ,10,165,15, 139,20
    )
445 CALL SCREEN(3)
446 CALL SOUND(-50,262
    ,10,196,15, 167,20
    )
447 CALL SCREEN(15)
448 CALL SOUND(-50,294
    ,10,220,15, 185,20
    )
449 NEXT REP
450 CALL SCREEN(2)
451 M$="les"&P$&"sont
    vainqueurs"
452 GOSUB 457
453 CALL HCHAR(24,3,32
    ,28)
454 M$="pour rejouer a
    ppuyez sur o"
455 GOSUB 457
456 GOTO 472
457 FOR I=1 TO 4
458 CALL SOUND(10,1946
    ,11)
459 NEXT I
460 FOR I=1 TO LEN(M$)
461 CALL HCHAR(24,2+I,
    ASC(SEG$(M$, I,1))
    )
462 NEXT I
463 FOR I=1 TO 100
464 NEXT I
465 RETURN
466 FOR I=1 TO 12
467 CALL COLOR(I,10,1)
468 NEXT I
469 M$="partie annulee
    "
470 GOSUB 457
471 GOSUB 453
472 CALL KEY(3,T,E)
473 IF E=0 THEN 472
474 IF T<>79 THEN 479
475 CALL HCHAR(24,3,32
    ,28)
476 P1=48
477 P2=48
478 GOTO 134
479 CALL CLEAR
480 IF G1=G2 THEN 484
481 IF G1>G2 THEN 486
482 PRINT "les pions r
    ouges gagnent"
483 GOTO 487
484 PRINT "partie null
    e"
485 GOTO 487
486 PRINT "les pions v
    erts gagnent"
487 END
488 IF RE=1 THEN 491
489 IF J=1 THEN 496
490 IF J=0 THEN 493
491 J=INT((10-1+1)*RND
    )+1
492 IF J>5 THEN 496
493 J=1
494 A=40
495 GOTO 498
496 J=0
497 A=88
498 CALL HCHAR(13,16,A
    )
499 RE=0
500 GOTO 152
```



Pointeurs et variables dynamiques en Pascal

Gérard Santraille

Nous avions vu, lors du précédent numéro, comment structurer des données en fichiers. Les divers types de fichiers manipulables par le PASCAL sont très riches et très nombreux, mais les tableaux, les articles ou les ensembles ont une taille fixée de façon définitive lors de l'écriture du programme. Avec le type **pointeur**, que nous allons examiner aujourd'hui, il est possible de construire des structures de données internes (c'est-à-dire en mémoire, sans accès aux divers dispositifs de stockage de masse comme les lecteurs de disquettes) de taille indéterminée.

L'avantage par rapport au tableau est immédiat. Pour pouvoir manipuler les éléments d'un tableau, il est nécessaire de déclarer sa taille au début du programme. Si l'on connaît le nombre exact de ses éléments on ne "gaspille" pas inutilement la mémoire (celle du TI est précieuse car rare) ; mais dans la grande majorité des applications on ne sait pas au début de l'exécution le nombre d'éléments qui seront manipulés. Bien souvent, pour s'affranchir de ce problème, on "surdimensionne" le tableau réservant ainsi de la mémoire non nécessairement utilisée.

Les pointeurs mettent en jeu des structures **extensibles** qui sont composées d'éléments créés **dynamiquement** lors de l'exécution. Ces éléments ne sont pas directement référencés par un indice, comme dans un tableau, mais sont désignés (pointés) par une variable pointeur.

Variables statiques et variables dynamiques

Une variable statique est déclarée dans un programme ou dans une procédure de façon univoque. Lors de l'exécution le programme affecte un emplacement mémoire à cette variable (la quantité de mémoire utilisée dépend du type de cette dernière). Que cette variable soit effectivement utilisée ou non son emplacement est définitif et ne peut par conséquent être ré-affecté pour une autre grandeur. On ne peut donc pas, à l'aide de variables statiques définir des structures de taille indéterminée.

Les variables dynamiques sont créées en cours d'exécution par l'instruction **new** dont nous verrons plus loin la syntaxe exacte. Il n'est pas possible lors de l'écriture du programme de leur donner un nom. Ces variables n'apparaîssent donc dans aucune déclaration du programme.

Variable pointeur

Les variables dynamiques sont désignées (pointées) par des variables (statiques ou dynamiques) de type pointeur. Le type pointeur n'a de sens que par rapport à un type déjà défini. Le point commun entre une variable pointeur et l'ensemble des variables qui peuvent être désignées par ce pointeur est le type des variables pointées. Une variable pointeur ne peut repérer que des variables dynamiques de même type.

Relisez bien ce paragraphe, il est fondamental.

La désignation d'un type pointeur se fait par le symbole "^" suivi du type des variables qu'il va pointer.

Ainsi, dans l'exemple suivant :

Voir programme 1

Programme 1

```
type

PERSONNE = record
              nom      : string[20];
              prenom   : string[15];
              tel      : string[15];
              adr1     : string[20];
              adr2     : string[20]
              end;

PTPERS     = ^PERSONNE

var

P1,P2 : PTPERS;
```

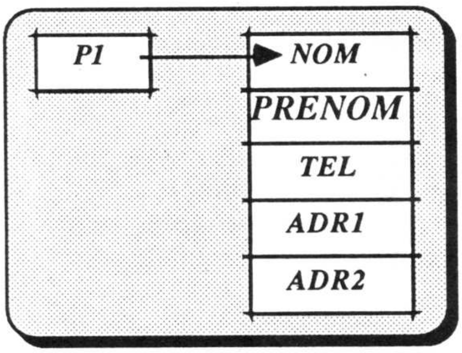
PERSONNE est un type article quelconque composé de CINQ enregistrements (NOM, PRENOM, TEL, ADR1, ADR2)
PTPERS est le type des variables pointeurs des objets de type PERSONNE.

P1 et P2 représentent des variables pointeurs (que nous appellerons simplement pointeurs) de type PTPERS.

Remarques

- P1 et P2 ne sont pas initialisés par cette déclaration.
- Aucune variable de type PERSONNE n'est créée par la déclaration de P1 ou P2.

Lorsque, lors de l'exécution, des variables de type PERSONNE seront créées, elles pourront être manipulées au moyen des variables P1 et P2.



Désignation des variables pointées

Une variable pointée est désignée de la même façon que le tampon ou la fenêtre de fichier (voir 99 Magazine numéro 8).

Ainsi, si P1 est un pointeur sur le type PERSONNE, **P1^** est une variable (dynamique) de type personne. **Cette variable ne doit en aucun cas être déclarée.**
P1^ est une variable à part entière de type PERSONNE dont on peut très classiquement manipuler les divers champs selon la syntaxe <u>P1^.champ</u>

P1^.nom représente donc le champ **nom** (de type string[20]) de la variable P1^ de type PERSONNE.

Opérations sur les pointeurs

Nous avons déjà vu que la déclaration P1,P2 : PTPERS pour reprendre l'exemple précédent n'initialisait en aucune façon P1 ou P2.
Pour activer un pointeur, on utilise l'instruction **new (pointeur)**.

Ainsi, dans l'exemple suivant :

```
type toto      = record
                 .............
                 ............
                 end;

var P : ^toto;

begin
new(P)
..........
```

la commande new(P) alloue dans la mémoire libre une variable de type toto, P pointe sur cette variable qui sera désignée par P^.

Remarque :

si toto est un type article avec variantes comme :

Voir programme 2

il convient de déclarer la valeur de toutes les variantes lors du new.
Le type toto possède une variante : selon la valeur du champ **rel** l'article comporte deux champs supplémentaires (societe et position) ou rien du tout.
Ainsi, **new(P,prof)** réservera en mémoire une variable dynamique de type toto avec la variante prof. Notons au passage que la taille effectivement réservée dépend des variantes sélectionnées et n'est pas la plus grande de ces dernières comme c'est le cas pour une allocation statique (d'où perte de place).

Les opérations possibles sur les pointeurs sont au nombre de trois.

- affectation
 1 - égalité de deux pointeurs de même type
 2 - pointeur : = **nil**; le pointeur ne repère alors aucune variable.
- logique
 3 - on peut tester l'égalité ou l'inégalité de deux pointeurs.

Premier exemple d'utilisation des pointeurs

Supposons que l'on dispose d'un fichier d'adresses recensant toutes nos relations. Chaque personne est définie par un article dont l'un des champs est une variable booléenne (vraie ou

Programme 2

```
type  relation   = (prof,perso);
      toto       = record
                   nom        : string[20];
                   prenom     : string[15];

                   case rel   : relation of
                   prof       : (societe : string[12];position : string[10]);

                   perso      : ()
```

fausse) décrivant une caractéristique donnée. Si nous désirons, pour un traitement ultérieur, charger en mémoire toutes les personnes dont ce champ est vrai nous disposons de plusieurs techniques.

1 - sans faire appel aux pointeurs

Si **personne** est le type de l'article et **caract** est le nom de la variable booléenne on peut lire tous les enregistrements du fichier et charger dans un tableau tous les enregistrements dont caract est vrai.

Comme on ne connaît pas à priori le nombre de personnes susceptibles d'avoir le champ caract vrai, il convient de surdimensionner le tableau destinataire.

Le programme pourrait être :

Voir programme 3

Programme 3

```
program NOPOINT;
type
        personne = record
        ................
        caract : boolean;
        ................
        end;
var
        i           : integer;
        fichier     : file of personne;
        tableau     : array [1..100] of personne;
begin
reset(fichier);
i:=0;
while not eof(fichier) do
        if fichier^.caract then   (* sous entendu vrai *)
        begin
        i:=i+1;
        tableau[i]:=fichier^
        end;
get(fichier)
end
end
```

2-En utilisant les pointeurs

Au lieu de surdimensionner à 100 le tableau destinataire des éléments de type personne on crée un tableau de pointeurs (ce qui prend beaucoup moins de place)

Voir programme 4

Programme 4

```
program POINT;
type
        personne = record
        ................
        caract : boolean;
        ................
        end;
        pointeur = ^personne;
var
        i           : integer;
        fichier     : file of personne;
        tableau     : array [1..100] of pointeur;
begin
reset(fichier);
i:=0;
while not eof(fichier) do
        if fichier^.caract then
        begin
        i:=i+1;
        new(tableau[i]);
        tableau[i]^:=fichier^
        end;
get(fichier)
end
end.
```

Attention à ne pas confondre un tampon de fichier (fichier^) avec une variable pointée (tableau[i]^).

Dans le premier programme chaque personne ayant le champ **caract** vrai se trouve physiquement dans **tableau**. Dans le second, **tableau[i]** est un pointeur et **tableau[i]^** est la variable pointée (de type personne) correspondante.

L'espace mémoire ainsi sauvegardé est très important. En effet, la taille du tableau des pointeurs additionnée à celle des variables dynamiques de type personne est bien inférieure à la taille d'un tableau de type personne.

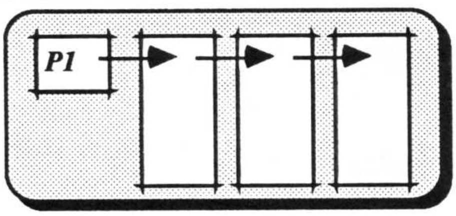
Remarque

Dans cet exemple nous avons utilisé un ensemble de variables statiques (le tableau des pointeurs) pour repérer un ensemble de variables dynamiques (les personnes ayant le champ caract vrai). En effet le tableau de pointeurs a été déclaré comme tel avant l'exécution du programme. Nous disposons d'autant de pointeurs que de variables pointées.

L'avantage de cette technique est que chaque variable dynamique est accessible **directement** puisqu'on dispose, pour chacune d'entre elles, de son pointeur (qui rappelons le est une variable statique). Son principal inconvénient est le même que pour le premier exemple : il est nécessaire de réserver tout un tableau (celui des pointeurs) qui ne sera pas nécessairement utilisé en totalité. Pour pallier cet inconvénient on peut imaginer de pointer les variables dynamiques par des pointeurs eux-mêmes créés dynamiquement.
Ces ensembles de données où les variables pointées se déduisent par récurrence s'appellent **structures chaînées** ou **listes**.

Structures de données récursives.

On peut imaginer un ensemble de variables dynamiques ayant chacune un champ constitué par un pointeur référencant la variable suivante.
On peut schématiser cette structure selon



où P1 est un pointeur qui désigne une première variable. Cette variable contient un pointeur qui permet d'accéder à la seconde, etc...

De telles variables doivent donc avoir la stucture suivante :

```
type  TOTO = record
             ............
             suivant : ^TOTO;
             ............
             end;
```

Un champ de l'article est donc un pointeur du type **auquel** il appartient.

En PASCAL cette écriture est normalement interdite puiqu'un article ne peut contenir un champ de son propre type. Une exception est faite pour les pointeurs. On doit d'abord définir le type des pointeurs sur les éléments de la liste, puis le type des éléments de la liste.
L'écriture correcte est donc:

```
type PTTOTO = ^TOTO
     TOTO   = record
              ............
              suivant : PTTOTO;
              ............
              end;
```

Une liste récursive simple dont chaque élément possède le pointeur du suivant n'admet qu'un seul mode d'accès. Une fois la liste définie on ne peut remonter au premier élément qu'en parcourant tous les autres.

Il s'agit d'une gestion d'accès de type **LIFO** (de l'anglais **L**ast **I**n - **F**irst **O**ut, littéralement dernier entré - premier sorti) : on retombe sur l'éternel problème de la pile d'assiettes.

Pour mieux comprendre ce qui se passe exécutons le petit programme suivant:

Voir programme 5

Ce programme permet de saisir cinq chaînes de caractères au clavier, il les stocke dans les variables de la liste puis les restitue en ordre inverse.

Analysons les différentes instructions.
Aucun commentaire spécial sur la partie déclarative du programme. Le type pointeur est bien défini **avant** le type des variables qu'il

Programme 5

```
program liste;
        type       ptr          = ^element
                   element      = record
                   info         : string[10];
                   suivant      : ptr
         end;
var p, premier     : ptr;
    i              : integer;
    texte  : string[10];
begin
writeln ('Creation de 5 elements');
premier:=nil;
for i:=1 to 5 do
                   begin
                   new(p);
                   readln(texte);
                   p^.info:=texte
                   p^.suivant:=premier;
                   premier:=p
end;
writeln ('Lecture des elements');
p:=premier
while p<>nil do
                   begin
                   writeln(p^.info');
                   p:=p^.suivant
end
end.
```

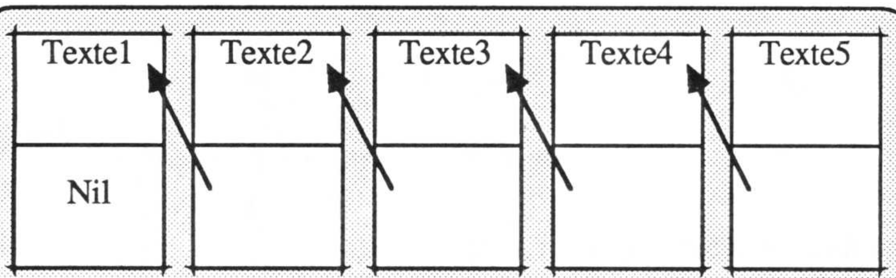


Schéma 1

va désigner. P et PREMIER sont des pointeurs (statiques puisque déclarés) qui serviront à parcourir la liste. SUIVANT est un champ de ELEMENT (variable dynamique) qui référençera la variable précédente.

L'instruction **premier:=nil;** initialise le pointeur premier (il ne pointe sur rien). L'instruction **new(p);** crée une variable dynamique p^ dont les deux champs sont définis par **p^.info:=texte** (chaîne lue au clavier) et **p^.suivant:= premier;**. La première variable dynamique (de type ELEMENT) contient donc un pointeur qui ne pointe sur rien. On affecte ensuite, par **premier:=p;** , à PREMIER la valeur de P qui, elle, pointe sur cette première variable que l'on vient de définir. Lors du second passage dans la boucle le champ 'suivant' de la seconde variable dynamique contiendra donc le pointeur référencant la première variable. De proche en proche on définit pour chacune des variables le champ 'suivant' comme référençant la variable qui la précède. La première variable n'ayant pas d'antécédent son pointeur vaut bien NIL.

Schématiquement on a :

Voir schéma 1

La suite du programme se contente de lire, dans le sens inverse de la définition, les différentes variables et affiche, pour chacunes d'entre elles le champ 'info'.
p:=premier; affecte bien à p la valeur du dernier pointeur (cette valeur n'est contenue dans aucune variable: il s'agit de la position de la cinquième variable). **writeln(p^.info)** affiche donc bien le champ info de la dernière variable. **p:= p^.suivant** affecte à **p** la position de la quatrième variable (qui est donc désormais **p^**), etc...

Il n'y a aucune limite à la complexité des structures que l'on peut ainsi construire. Il suffit de gérer plusieurs pointeurs dans les champs des éléments composant une liste.
Un exemple classique est la liste circulaire que l'on pourrait schématiser selon:

Voir schéma 2

Destruction des variables dynamiques

Contrairement aux variables statiques, la durée de vie des variables dynamiques est indépendante de la procédure dans laquelle elles ont été créées. Pour détruire ces variables il convient d'effectuer une opération explicite.
L'instruction **dispose(P)** désalloue la variable P^ : la valeur de P devient alors indéfinie.

Remarques

• Si la variable pointée est un article avec variante créée par **new(P,V1,V2)** il faut utiliser **dispose(P,V1,V2)** et avec les mêmes variantes que lors de sa création.
• La disparition de tout repère sur une variable pointée n'entraîne pas la désallocation de cette variable. Ainsi la séquence **new(P); P:=nil;** fait que la variable P^ créée lors du new ne sera plus jamais accessible alors que l'espace qui lui a été alloué encombrera la mémoire jusqu'à la fin du programme.
• On ne peut exécuter un dispose(P) si P est indéfini ou possède la valeur NIL.

Exemple d'utilisation des pointeurs

Le petit programme suivant illustre l'une des utilisations des pointeurs pour la création et la mise à jour d'un ensemble de données dynamiques.

Voir programme 6

Ce programme lit un fichier (texte quelconque ou source PASCAL) et établit la liste des mots utilisés ainsi que leur nombre d'apparition. Comme on ne peut savoir en début d'exécution le nombre de mots différents contenus dans le texte, ce programme crée au fur et à mesure un dictionnaire dynamique répertoriant chaque mot.

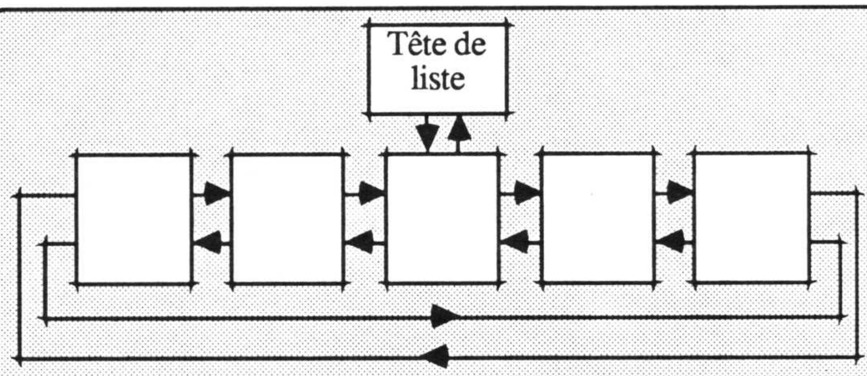


Schéma 2

Programme 6

Suite du programme page suivante ⇓

```
program POINTEURS;
const
        lmax=15;      {taille maxi des identificateurs}
type
        ptelem = ^elem;
        elem  = record
                suiv        : ptelem; {pointeur de la variable suivante}
                id          : string[15];
                n           : integer
        end;
        classe = (lettre,chiffre,quote,accolade,autre);
var
        fichier     : string[20];
        f           : file of char;
        categ       : array[char] of classe;
        ident       : string[15];   {identificateur courant}
        tete        : ptelem;       {tete de la liste}
        c           : char;          {caractere courant}
procedure fairecategorie;
var
        i           : 0..255;
        c           : char;
begin
        for i:=0 to 255 do categ[chr(i)]:=autre;
        for c:='A' to 'Z' do categ[c]:=lettre;
        for c:='a' to 'z' do categ[c]:=lettre;
        for c:='0' to '9' do categ[c]:=chiffre;
        categ['{']:=accolade;
        categ['}']:=accolade;
        categ['''']:=quote
end;
procedure impliste;
var
        p:ptelem;
begin
page(output);
writeln('':10,'Impression du dictionnaire');
writeln('':10,'__________________________');
writeln;
p:=tete;
        while p<>nil do
        with p^ do
                begin
                writeln('':10,id,'  ==>',n:4);
                p:=suiv
                end
        end;
```

Les variables dynamiques ainsi créées sont des articles sans variante (ELEM) ayant chacune trois champs:

- **suiv** qui est un pointeur référencant la variable suivante (il s'agit d'une liste où, comme nous l'avons vu, chaque variable pointe sur sa précédente)
- **id** qui est une chaîne de caractères représentant le mot à mémoriser
- **n** de type entier qui représente le nombre d'occurences du mot dans tout le texte.

Pour pouvoir traiter les fichiers sources écrits en PASCAL ce programme ignore les identificateurs qui apparaissent dans des commentaires (c-a-d entre {}).

Le programme se compose

- d'une procédure d'initialisation (FAIRECATEGORIE) qui définit le type de chaque caractère rencontré. Notez au passage l'une des originalités de Pascal qui permet de créer des tableaux dont les identificateurs des éléments ne sont pas entiers mais de type ordinal de base **char**. CATEG est un tableau dont les éléments sont de type "classe" (tous les composants de ce type sont donnés explicitement) et dont l'indexation se fait par un caractère.

- d'une procédure d'impression de la liste (du dictionnaire). Cette procédure ne pose aucun problème particulier : elle est parcourue séquentiellement en fin de programme et affiche à l'écran tous les mots avec le nombre d'apparitions.

• d'une procédure d'identification du mot en cours de lecture (LIREIDENT). Cette procédure ignore donc les commentaires et poursuit la lecture du fichier d'entrée jusqu'à la fin du mot lorsqu'elle rencontre un caractère.

• d'une procédure de recherche en liste (CHERCHEID). Cette procédure parcourt séquentiellement la liste pour voir si le mot courant a déjà été référencé. Si tel est le cas, le champ **n** de la variable en question est incrémenté de 1, sinon une nouvelle variable dynamique est créée. Le champ **n** est de cette variable est mis à 1, le champ **suiv** pointe la variable suivante (il s'agit chronologiquement de la variable précédente). Le pointeur de tête de liste (celui qui définit le premier élément du dictionnaire) référence alors cette nouvelle variable.

En fait, chaque fois que l'on crée une variable dynamique elle prend la première position de la liste et pointe sur l'ancienne variable de tête. On peut schématiser ce processus par la figure suivante.

Voir schéma 3

Les structures dynamiques chaînées sont très souples et très puissantes. Néanmoins l'accès direct à un élément donné est impossible : il est nécessaire de parcourir séquentiellement la liste jusqu'à sa rencontre. De même l'insertion ou la suppresion d'un élément **au milieu** d'une liste n'est pas très aisée puiqu'elle suppose la mise à jour des pointeurs.

Si le traitement que l'on doit faire sur un ensemble de données est global et ne nécessite pas l'accès direct ou sélectif à certains nombres de ses éléments l'emploi d'une liste est un excellent moyen pour sauvegarder de la mémoire.

Plus généralement l'emploi de variables dynamiques (pointées par des variables statiques ou dynamiques) est très intéressant dès qu'il s'agit de gérer de gros

Programme 6 (suite)

```
procedure lireident;
var
        k          : 0..lmax;
        trouve     : boolean;
begin
trouve:=false;
repeat
        case categ[c] of
        lettre : begin
        ident:='               '; {15 blancs}
         k:=0;
        repeat
                if k<lmax then
                begin
                k:=k+1;
                ident[k]:=c
                end;
        read(f,c);
        write(c);
        until (categ[c]<>lettre) and (categ[c]<>chiffre);
        trouve:=true;
        end;
        accolade:repeat
        read(f,c);
        write(c);
        until c='}';
        quote:repeat
```

Suite du programme page suivante ⇒

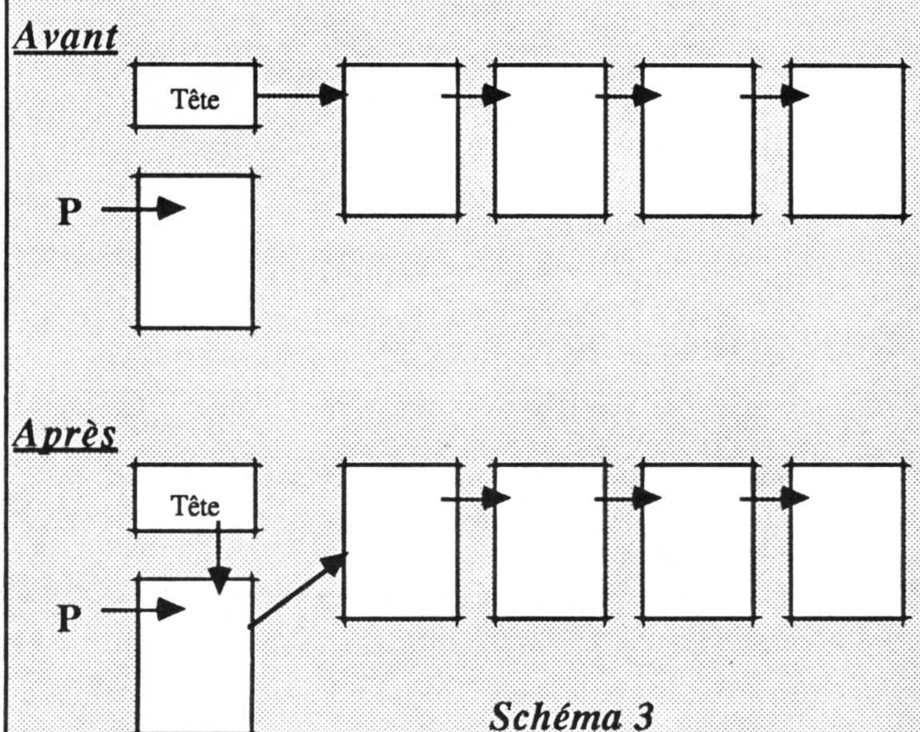


Schéma 3

Programme 6 (suite et fin)

```
          read(f,c);
          write(c);
          until c='';

          chiffre,autre:begin
          read(f,c);
          write(c)
          end

          end {fin du case}

until trouve or eof(f)
end;

procedure cherchid;

var
          pastrouve:boolean;
          p:ptelem;

begin
p:=tete;
pastrouve:=true;
while (p<>nil) and pastrouve do
          if p^.id=ident then begin
          pastrouve:=false;
          p^.n:=p^.n +1
          end
          else p:=p^.suiv;

if pastrouve then
          begin
          new(p);
          p^.id:=ident;
          p^.n:=1;
          p^.suiv:=tete;
          tete:=p;
          end
end;

begin   {programme principal)
page(output);
fairecategorie;
tete:=nil;
{$I-}
          repeat
          write('FICHIER ? ');
          readln(fichier);
          reset(f,fichier);
          until ioresult=0;
{$I+}
read(f,c);
write(c);
while not eof(f) do
          begin
          lireident;
          cherchid
end;
close(f);
impliste
end.
```

volumes d'information dont on ne connaît pas la taille à priori. La plus importante restriction de cette technique vient du fait que sur le TI aucune vérification de la taille de la mémoire disponible n'est faite lors de la création d'une nouvelle variable. C'est donc au programmeur de s'assurer (au moyen de MEMAVAIL, VARAVAIL, VARDISPOSE et VARNEW) de l'intégrité de sa partition mémoire.

99

99 MAGAZINE

vous attend à la boutique Sicob - Porte Maillot - au stand M279.

Musique et Mini-mémoire

Denise Amrouche

Si vous programmez souvent en Basic TI, vous vous êtes certainement heurtés au problème suivant : faire entendre de la musique pendant un jeu ou pendant l'installation d'un décor. Synchroniser les CALL SOUND et les autres instructions nécessite de nombreux tâtonnements.

Source de la routine "Musique et Mini-mémoire"

```
                  *RESERVER UNE PLACE EN VDP
                  *LONGUEUR DE LA LISTE DES SONS DANS >7118
                  *PREMIERE ADRESSE DE LA PLACE ALLOUEE DANS R0
7FBA                     AORG >7FBA
7FBA C820                MOV  @>7118,@>830C
7FBC 7118
7FBE 830C
7FC0 04E0                CLR  @>837C
7FC2 837C
7FC4 0420                BLWP @>6018        (GPLLNK)
7FC6 6018
7FC8 0038                DATA >0038
7FCA C020                MOV  @>831C,R0
7FCC 831C
                  *CHARGER EN MEMOIRE VIDEO  LA LISTE DES SONS
7FCE 0201                LI   R1,>711A  (1ERE ADRESSE DE LA LISTE)
7FD0 711A
7FD2 C0A0                MOV  @>7118,R2 (LONGUEUR DE LA  LISTE)
7FD4 7118
7FD6 0420                BLWP @>6028      (VMBW)
7FD8 6028
                  *LANCER L'EXECUTION
7FDA 0300                LIMI 0
7FDC 0000
7FDE C800                MOV  R0,@>83CC
7FE0 83CC
7FE2 0202                LI   R2,>0100
7FE4 0100
7FE6 D802                MOVB R2,@>83CE
7FE8 83CE
7FEA F802                SOCB R2,@>83FD
7FEC 83FD
7FEE 0300                LIMI 2
7FF0 0002
7FF2 04E0                CLR  @>837C
7FF4 837C
7FF6 045B                B    *R11      (RETOUR)
7FF8   4D                TEXT 'MUSIQU'
7FFE 7FBA                DATA >7FBA
                         END
```

Ce programme Basic va vous permettre d'écrire en langage machine, la liste des sons à émettre, et d'implanter la routine pour exécuter ces sons dans la mini mémoire. Vous pourrez alors, à tout moment, directement ou dans un programme, en tapant *CALL LINK ("MUSIQU")*, faire exécuter votre morceau.

Toute la partie "DATA" du programme représente les notes à jouer. Elles sont fournies à peu près comme en Basic.

- durée multipliée ensuite par M (ici M = 167) à diminuer pour un rythme plus rapide ;
- fréquence, puissance du canal 1 ;
- fréquence, puissance du canal 2 ;
- fréquence, puissance du canal 3.

Différences par rapport à CALL SOUND

Quelques différences par rapport au *CALL SOUND* du Basic : chaque ligne de "DATA" doit comporter 6 virgules ; sur un canal donné, une note commencée ne s'arrête que si une autre note est envoyée (éventuellement avec une puissance 30 si on ne veut pas l'entendre). On peut donc ne rien mettre comme fréquence et puissance si l'on veut réentendre la même note sur le même canal. La dernière "DATA" est par convention DATA O pour terminer le processus.

Utilisation

Taper le programme Basic (les lignes de "DATA" peuvent être modifiées suivant l'air que vous voulez entendre).

Le sauvegarder, car d'une manière générale, lorsqu'un programme fait appel à une routine en assembleur, on risque l'erreur fatale.

Faire RUN et... patienter. On implante ainsi en fin de mini mémoire, la routine assembleur listée et commentée en fin d'article, puis la liste des sons codée comme il faut pour l'exécution.

Le programme indique s'il y a trop de "DATA" pour la capacité de la mini mémoire, ou le nombre d'octets occupés (3746 au maximum).

La liste des sons et la routine étant maintenant en mini mémoire, CALL LINK ("MUSIQU") provoquera à tout moment l'exécution (en occupant une partie de la mémoire vidéo). CALL SOUND (-1, 400, 30), par exemple, provoquera si on le souhaite une interruption avant la fin.

99

Programme de démonstration de la routine "Musique et Mini-mémoire"

```
100 REM ECRITURE DE
    MUSIQUE EN MINIM
    EM
110 REM ******PAR DENIS
    E AMROUCHE *******
120 REM CHARGEMENT
    EN MINI-MEMOIRE D
    E LA ROUTINE ASS
    EMBLEUR
130 CALL LOAD(32698,20
    0,32,113,24,131,12
    ,4,224)
140 CALL LOAD(32706,13
    1,124,4,32,96,24,0
    ,56)
150 CALL LOAD(32714,19
    2,32,131,28,2,1,11
    3,26)
160 CALL LOAD(32722,19
    2,160,113,24,4,32,
    96,40)
170 CALL LOAD(32730,3,
    0,0,0,200,0,131,20
    4)
180 CALL LOAD(32738,2,
    2,1,0,216,2,131,20
    6)
190 CALL LOAD(32746,24
    8,2,131,253,3,0,0,
    2)
200 CALL LOAD(32754,4,
    224,131,124,4,91)
210 REM MISE DU NOM
    DFDE LA ROUTINE
220 CALL LOAD(32760,77
    ,85,83,73,81,85,12
    7,182)
230 CALL LOAD(28700,12
    7,248,127,248)
240 REM ----------------------
-   -----------------------
250 CALL CLEAR
260 REM LISTE DES SO
    NS:DUREE,FREQUEN
    CE1,PUISSANCE1,FRE
    QU2,PUIS2,FREQU3,P
    UIS3
270 REM LA DUREE EST
    MULTIPLIEE PAR M
280 M=167
290 REM UNE NOTE PAR
    LIGNE,6 VIRGULES P
    AR LIGNE
300 DATA 1,466,0,,,,
310 DATA 1,466,0,,,,
320 DATA 2,523,0,156,0
    ,,
330 DATA 2,466,0,196,0
    ,,
340 DATA 2,466,0,233,0
    ,,
350 DATA 2,466,0,156,0
    ,,
360 DATA 2,392,0,196,0
    ,,
370 DATA 2,392,0,233,0
    ,,
380 DATA 2,349,0,156,0
    ,,
390 DATA 2,311,0,196,0
    ,,
400 DATA 2,392,0,233,0
    ,,
410 DATA 2,466,0,156,0
    ,,
420 DATA 2,466,0,196,0
    ,,
430 DATA 2,466,0,233,0
    ,,
440 DATA 2,262,0,131,0
    ,,
450 DATA 2,311,0,156,0
    ,,
460 DATA 2,311,0,196,0
    ,,
470 DATA 2,349,0,131,0
    ,,
480 DATA 2,311,0,156,0
    ,,
490 DATA 2,349,0,196,0
    ,,
500 DATA 2,392,0,156,0
    ,,
510 DATA 2,466,0,196,0
    ,,
520 DATA 2,466,0,233,0
    ,,
530 DATA 2,466,0,156,0
    ,,
540 DATA 2,466,0,196,0
    ,,
550 DATA 2,466,0,233,0
    ,,
560 DATA 2,523,0,156,0
    ,,
570 DATA 2,466,0,196,0
    ,,
580 DATA 2,466,0,233,0
    ,,
590 DATA 2,466,0,156,0
    ,,
600 DATA 2,392,0,196,0
    ,,
610 DATA 2,392,0,233,0
    ,,
620 DATA 2,349,0,156,0
    ,,
630 DATA 2,311,0,196,0
    ,,
640 DATA 2,392,0,233,0
    ,,
650 DATA 2,466,0,156,0
    ,,
660 DATA 2,466,0,196,0
    ,,
670 DATA 1,466,0,233,0
    ,,
680 DATA 1,466,0,233,0
    ,,
690 DATA 2,262,0,131,0
    ,,
700 DATA 2,311,0,156,0
    ,,
710 DATA 2,311,0,196,0
```

```
  ,,
720 DATA 2,349,0,131,0
  ,,
730 DATA 2,311,0,156,0
  ,,
740 DATA 2,349,0,196,0
  ,,
750 DATA 2,392,0,156,0
  ,,
760 DATA 2,311,0,196,0
  ,,
770 DATA 2,349,0,233,0
  ,,
780 DATA 2,311,0,156,0
  ,,
790 DATA 2,311,0,196,0
  ,,
800 DATA 2,311,0,233,0
  ,,
810 DATA 2,262,0,131,0
  ,,
820 DATA 2,311,0,156,0
  ,,
830 DATA 2,311,0,196,0
  ,,
840 DATA 2,262,0,131,0
  ,,
850 DATA 2,311,0,156,0
  ,,
860 DATA 2,311,0,196,0
  ,,
870 DATA 2,294,0,147,0
  ,,
880 DATA 2,311,0,156,0
  ,,
890 DATA 2,311,0,156,0
  ,,
900 DATA 2,466,0,196,0
  ,,
910 DATA 2,466,0,196,0
  ,,
920 DATA 1,466,0,233,0
  ,,
930 DATA 1,466,0,233,0
  ,,
940 DATA 2,262,0,131,0
  ,,
950 DATA 2,311,0,156,0
  ,,
960 DATA 2,311,0,196,0
  ,,
970 DATA 2,349,0,131,0
  ,,
980 DATA 2,311,0,156,0
  ,,
990 DATA 2,349,0,196,0
  ,,
1000 DATA 2,392,0,156,
     0,,
1010 DATA 2,466,0,196,
     0,,
1020 DATA 2,466,0,233,
     0,,
1030 DATA 2,466,0,156,
     0,,
1040 DATA 2,466,0,196,
     0,,
1050 DATA 1,466,0,233,
     0,,
1060 DATA 1,466,0,233,
     0,,
1070 DATA 2,523,0,156,
     0,,
1080 DATA 2,466,0,196,
     0,,
1090 DATA 2,466,0,233,
     0,,
1100 DATA 2,466,0,156,
     0,,
1110 DATA 2,392,9,196,
     0,,
1120 DATA 2,392,0,233,
     0,,
1130 DATA 2,349,0,156,
     0,,
1140 DATA 2,311,0,196,
     0,,
1150 DATA 2,392,0,233,
     0,,
1160 DATA 2,466,0,156,
     0,,
1170 DATA 2,466,0,196,
     0,,
1180 DATA 2,466,0,233,
     0,,
1190 DATA 2,262,0,131,
     0,,
1200 DATA 2,311,0,156,
     0,,
1210 DATA 2,311,0,196,
     0,,
1220 DATA 2,349,0,131,
     0,,
1230 DATA 2,311,0,156,
     0,,
1240 DATA 2,349,0,196,
     0,,
1250 DATA 2,392,0,156,
     0,,
1260 DATA 2,311,0,196,
     0,,
1270 DATA 2,349,0,233,
     0,,
1280 DATA 4,311,0,196,
     0,,
1290 DATA 2,311,0,196,
     0,,
1300 DATA 2,311,0,196,
     30,,
1310 DATA 0
1320 REM LA LISTE SE
     TERMINE TOUJOUR
     S PAR DATA 0
1330 DE=28954
1340 AD=28954
1350 READ D
1360 IF (D<0)+(D>4250/
     M)THEN 1710
1370 IF D=0 THEN 1630
1380 ADR=AD+1
1390 FOR V=1 TO 3
1400 READ FR$,PU$
1410 IF PU$="" THEN 15
     10
1420 F=VAL(FR$)
1430 P=VAL(PU$)
1440 IF (F<110)+(F>447
     33)THEN 1710
1450 IF (P<0)+(P>30)TH
     EN 1710
1460 G=111860.8/F
1470 G2=INT(G/16)
1480 G1=G-16*G2
1490 CALL LOAD(ADR,96+
     32*V+G1,G2,112+32
     *V+P/2)
1500 ADR=ADR+3
1510 NEXT V
1520 D=D*M*6/100
1530 IF D>1 THEN 1550
1540 D=1
1550 CALL LOAD(ADR,D)
1560 ADR=ADR+1
1570 DIF=ADR-AD-2
1580 CALL LOAD(AD,DIF)
1590 PRINT I
1600 I=I+1
1610 AD=ADR
1620 IF AD>32780 THEN
1730 ELSE 1350
1630 CALL LOAD(AD,4,15
     9,191,223,255,0)
1640 BY=AD+6-DE
1650 BY1=INT(BY/256)
1660 CALL LOAD(28952,B
     Y1,BY-256*BY1)
1670 PRINT "NOMBRE DE
     BYTES UTILISES :"
     ;BY+2
1680 CALL LINK("MUSIQU
     ")
1690 STOP
1700 REM MESSAGE D'
     ERREUR DANS LES
     DATA
1710 PRINT "ERREUR DAN
     S LA DATA NO";I+1
1720 STOP
1730 PRINT "JE NE PEUX
      PLUS CHARGER AU
     DELA DE LA DATA N
     O :";I
1740 PRINT "CAR JE N'A
     I PLUS DE PLACE D
     ANS LA MINI-MEMOI
     RE"
```

Transfert Image/Cassette

Gérard Baroni et Jean Marin

Après vous avoir indiqué brièvement comment, avec la norme Texas Instruments, un transfert d'image vers une cassette - ou inversement - peut s'effectuer, nous exposerons notre solution qui apporte un gain substantiel de temps, de place mémoire et qui, de plus, donne accès à tous les modes graphiques du TI-99/4A.

La configuration exigée comprend la Mini-mémoire ou le Basic étendu plus l'extension 32Ko.

Voyons d'abord la manière de procéder en utilisant la norme Texas.

- En mode standard, il s'agit de réaliser une copie d'écran et de sauvegarder les codes ASCII des 768 cases de l'écran. Pour le transfert de ces 768 octets vers la cassette à l'aide de OPEN # CS1 et PRINT #..., 192 octets sont transférés toutes les 10 secondes. 40 secondes sont nécessaires pour expédier le contenu de l'écran vers notre cassette.
- Pour sauvegarder les tables d'écrans, des couleurs et des formes, soit 2048 octets au maximum, 11 enregistrements sont nécessaires et le transfert dure environ 2 minutes.
- En BitMap, où nous avons 14 Ko d'image et 2 Ko de Basic restant, il y a interférence entre les "buffers" cassette et le reste. Avec 75 enregistrements de 10 secondes, il faudrait environ 12 minutes pour transférer l'image.

Notre solution

Pour disposer d'une plus grande liberté, nous allons nous brancher sur les routines ROM de la console sans passer par les routines GPL qui sont dans les GROMS. Les avantages sont les suivants :

1) Le temps de silence pour l'amorce (avant le sifflement) disparait.
2) Le transfert direct de l'image vers la cassette - et inversement - se fait sans passer par un buffer. Donc, il n'y a pas de place perdue en mémoire. A la lecture, l'image apparait au fur et à mesure du déroulement de la cassette.

On peut aller *en un seul enregistrement* jusqu'à un maximum de 255 fois 64 octets soit 16 320 octets, c'est à dire la totalité de la mémoire vive vidéo moins 64 octets.

Temps de tranfert

- Table écran (768 octets) :
 15 secondes au lieu de 40 ;
- Image (2048 octets) :
 30 secondes au lieu de 120 ;
- Image BitMap (14Ko) :
 3'15" au lieu de 12'30".

Les programmes "SI" (Sauve Image) et "LI" (Lire Image), en assembleur dont les listings suivent, sont accessibles du Basic par des CALL LINK("SI") ou ("LI").

Après exécution, le retour au Basic se fait automatiquement, étant prévu dans la routine console.

Important

Dans le mode graphique 2 Ko, le CALL LINK ("LI") doit être précédé de CALL CHAR (159,"") qui a pour effet de réserver 2 Ko d'image.

En BitMap, le CALL LINK ("LI") doit être précédé d'un CALL LOAD (-31890,56,0) qui réserve 14 Ko d'image et d'un CALL LINK ("BITMAP") - *voir 99 magazine précédent* - qui initialise le mode graphique BitMap dans le processeur vidéo.

Méthode de transfert employée par Texas Instruments

Les programmes "LI" et "SI" réalisent des transferts d'informations, mais étudions un peu le mécanisme de transfert prévu par le constructeur.

Nous savons que tout message est constitué par une suite d'informations élémentaires codées sous forme binaire (0, 1). Ainsi, lors de l'enregistrement sur cassette ("SAVE CS1"), chacune de ces informations sera représentée par un signal d'une durée déterminée permettant à la machine de distinguer, au moment de la lecture ("OLD CS1"), entre la valeur 0 ou 1 du bit enregistré. Dans le système Texas, les mesures effectuées à l'aide d'un oscilloscope nous donnent les résultats suivants :

- lorsqu'il s'agit du bit de valeur 0, la durée (t) du signal est approximativement de 0,74 milliseconde (ms) ;
- dans le cas du bit de valeur 1, (t) correspond à 2 phases d'émission séparées par un pic d'environ 0,37 ms chacune (t/2). La fréquence du signal enregistré est de l'ordre de 1400 bits/sec ou 10 Ko/minute.

Examinons maintenant le contenu de l'enregistrement d'un programme sur cassette. Au début de celui-ci, nous entendons un sifflement pendant plus de 14", ce qui équivaut au passage de 768 octets de valeur 0. Puis, survient 1 octet de valeur >FF annonçant le début d'un message, puis 2 octets identiques de valeur N (N désigne le nombre de tranches de 64 octets qui vont constituer le message proprement dit ; ce nombre varie en fonction de la longueur du programme).

Tout ceci précède N groupes de 74 octets répétés 2 fois. Dans chacun de ces groupes, les octets se répartissent ainsi :

- 8 octets à 0 ;
- 1 octet à >FF ;
- 1 paquet de 64 octets correspondant au message ;
- 1 octet "checksum", ou somme modulo 256, des 64 octets permettant à la machine de tester la présence ou non d'erreurs dans un groupe.

Les 8 octets de valeurs 0 et l'octet de valeur >FF sont très important : ils mesurent la vitesse du magnétophone et permettent une adaptation systématique constituant un véritable asservissement de vitesse, et démontrent la haute fiabilité du système.

Le "checksum" est un octet dont la valeur est calculée à l'enregistrement. Lors de la lecture, cette valeur sera comparée et devra être égale au reste de la division entière par 256 de la somme des valeurs des 64 octets. Comme nous savons que la valeur d'un octet ne saurait dépasser 255, si la somme des valeurs des 64 octets est 260 par exemple, le "checksum" devra être égal à 4 (260 modulo 256 = 4).

La répétition des groupes de 74 octets permet donc de corriger une éventuelle erreur. En effet, lorsqu'un bit est faux, le "checksum" ne correspondra pas à la somme modulo 256 et, par conséquent, l'ordinateur saura que cette tranche est erronée. C'est pourquoi il prendra l'autre tranche en totalité, à condition qu'elle soit bonne. Dans le cas contraire, il affichera "DATA ERROR". Par analogie, on peut comparer le TI à un véhicule disposant d'une roue de secours, qui sera utilisée lors d'une crevaison.

Nous voyons combien ce dispositif ralentit la transmission des données puisqu'il y a en fait 148 octets pour un message réel de 64 octets seulement. La vitesse moyenne est de 600 bits/seconde, soit 75 octets/seconde, soit 4,5 Ko/minute.

Les gens pressés peuvent modifier les routines en ROM, après translation dans la RAM de la Mini-mémoire, pour supprimer la duplication des tranches, réalisant ainsi un gain de temps appréciable (t/2). Ils peuvent même accélérer le signal dans un rapport 2 si le magnétophone le permet.

On arrive ainsi à transférer une image BitMap de 14 Ko en 35 secondes environ. La même technique permet aussi de charger l'extension mémoire rapidement.

Source des routines SI" et "LI"

```
--------------------------------------------
"SI"
                 AORG  >XXXX  * Adresse départ
                              * à choisir
XXXX  02E0       LWPI  >83E0
      83E0
      0201       LI    R1,TA
      XXXX
      0460       B     @>1346 * Vers routine
      1346
--------------------------------------------
XXXX  XXXX  TA   DATA  N      * Nombre d'octets
                              * à transférer
      XXXX       DATA  A      * Adresse de base
                              * en RAM vidéo
                              * (0 : normal)
--------------------------------------------
"LI"
XXXX  02E0       LWPI  >83E0
      83E0
      0201       LI    R1,TA
      XXXX
      0460       B     @>142E
--------------------------------------------
XXXX  XXXX  TA   DATA  N
      XXXX       DATA  A
--------------------------------------------
NE PAS OUBLIER LA TABLE REF/DEF
--------------------------------------------
```

Votre assiduité méritant récompense, voici joint le listing d'un programme éditeur de dessin en BitMap qui vous permettra de créer et de sauvegarder sur cassette vos graphismes. Ce programme utilise les routines vues dans les numéros précédents de 99 magazine.

NB : En cas de modification du programme, souvenez-vous que vous ne disposez que de 2 Ko pour le Basic et que les REMs et les noms de variables trop longs sont à proscrire.

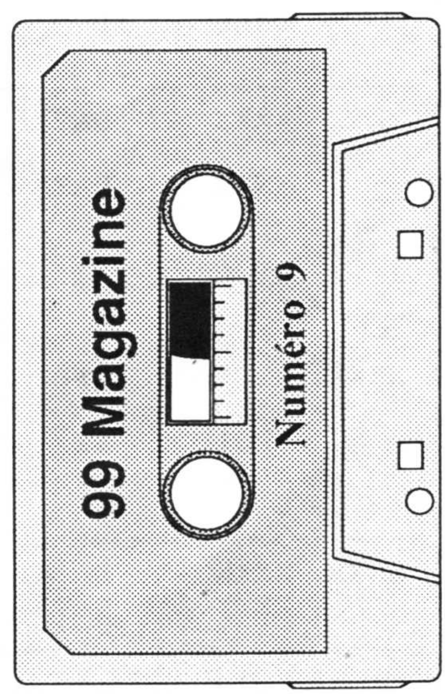
99

Routines "SI" et "LI"

Programme de démonstration

Basic TI et Mini-mémoire

```
100 CALL LOAD(-31890,0
    )
110 CALL CLEAR
120 PRINT " LES COMMAN
    DES SONT:"::"-D,S,
    E,X POUR DEPLACER"
    :"          LE CURS
    EUR"::
130 PRINT "-R         DE
    PLACEMENT RAPIDE":
    :"-L        DEPLACE
    MENT LENT"::
140 PRINT "-C         CO
    ULEUR"::"-BEGIN
    DEPART DU SEGMENT"
    :"          A TRACE
    R"::"-ENTER    TRAC
    E DU SEGMENT"::
150 PRINT "-AID       SA
    UVER SUR K7"::"-BA
    CK    RETOUR AU BA
    SIC"::
160 INPUT "(ENTER)  PO
    UR COMMENCER":R$
170 CALL CLEAR
180 PRINT "VOULEZ-VOUS
     RETROUVER UN"::"D
    ESSIN SUR K7 ";
190 INPUT "(O/N)? ":R$
200 CALL LINK("BITMAP"
    )
210 IF R$<>"O" THEN 23
    0
220 CALL LINK("LI")
230 D=1
240 C=240
250 Y=96
260 X=128
270 CALL POKEV(6912,Y-
    3,X-3,128,C/16,208
    ,"",7168,0,68,56,4
    0,56,68,0,0)
280 YD=Y
290 XD=X
300 CALL POKEV(6912,Y-
    3,X-3,128,C/16)
310 CALL KEY(5,R,E)
320 IF E=0 THEN 310
330 IF R<>69 THEN 370
340 Y=Y-D
350 Y=Y-D*(Y<0)
360 GOTO 300
370 IF R<>88 THEN 410
380 Y=Y+D
390 Y=Y+D*(Y>191)
400 GOTO 300
410 IF R<>83 THEN 450
420 X=X-D
430 X=X-D*(X<8)
440 GOTO 300
450 IF R<>68 THEN 490
460 X=X+D
470 X=X+D*(X>255)
480 GOTO 300
490 IF R<>67 THEN 520
500 C=C+16+240*(C=240)
510 GOTO 300
520 IF R<>14 THEN 560
530 XD=X
550 YD=Y
560 IF R<>13 THEN 610
570 CALL LINK("COLOR",
    C)
580 CALL LINK("DROITE"
    ,YD,XD,Y,X)
590 CALL SOUND(50,440,
    0)
600 GOTO 280
610 IF R<>82 THEN 650
620 CALL SOUND(50,1000
    ,0)
630 D=8
640 GOTO 310
650 IF R<>76 THEN 690
660 CALL SOUND(50,1000
    ,0)
670 D=1
680 GOTO 310
690 IF R<>1 THEN 720
700 CALL LINK("SI")
710 GOTO 310
720 IF R<>15 THEN 310
730 CALL PEEKV(-32768,
    X)
```



Starline

Georges Goument

Nous vous proposons ici un jeu en trois dimensions, pour deux joueurs. Vous pouvez évidemment jouer seul, mais cela n'a pas grand intérêt, sinon celui de se familiariser avec le maniement du clavier et s'entrainer.

Règles du jeu

Chaque joueur dispose de douze étoiles réparties sur quatre plateaux. Pour gagner une partie, il faut être le premier à réaliser un alignement de quatre étoiles. Mais attention, pas n'importe quel alignement... vous devez retrouver celui choisi par l'ordinateur parmi les 68 possibilités offertes.

Les déplacements en diagonale sont interdits, vous devez toujours suivre lignes ou colonnes. Le programme refusera votre jeu si vous cherchez à tricher.

Vous pouvez également changer de plateau pour monter ou descendre vos étoiles, à condition que cette opération respecte la règle suivante : la nouvelle position doit être située immédiatement à la verticale de l'ancienne, et libre de toute étoile.

Il est possible de gêner considérablement le jeu adverse puisque les permutations d'étoiles entre joueurs sont réalisables. Ce système permet d'employer plusieurs tactiques de jeu. A vous de les découvrir pour les utiliser au bon moment, et remporter la partie. Les permutations sont impossibles quand on change de plateau (le jeu serait trop facile).

Chaque étoile bien placée donne un "bip" sonore aigü.

Jeu à partir du clavier

Touches 1 à 4 (plateau)

Numéro de plateau. Ces numéros changent de couleur selon le tour du joueur (fond rouge ou blanc).

Touches 1 à 4 (étoile)

Coordonnées de l'étoile. Numéro de ligne en premier (sur la gauche du plateau) puis numéro de colonne (chiffres devant le plateau). L'étoile change de couleur.

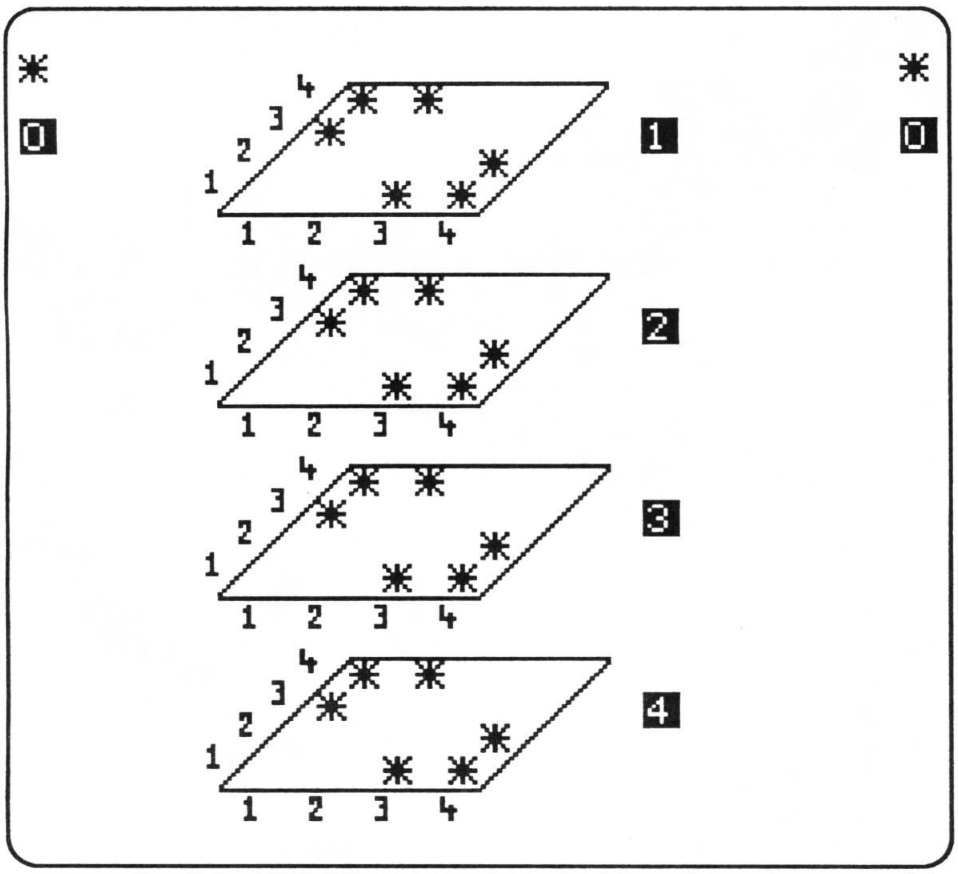
Touches 1 à 4 (coordonnées)

Tapez les nouvelles coordonnées : 1 à 4. Si vous ne changez pas de plateau de jeu, l'étoile occupe sa nouvelle position ou permute avec une étoile adverse.

Touche N

Changement de plateau : touche N, puis 1 à 4 = nouveau numéro de plateau ; plus 1 à 4 pour la nouvelle position.
La touche N sert également pour l'annulation d'un coup, afin de jouer une autre position ou un autre plateau.

99



Programme Starline

Basic TI

```
10 REM ***************
11 REM *             *
12 REM *  STARLINE   *
13 REM *             *
14 REM *  Basic  TI  *
15 REM *             *
16 REM *  copyright  *
17 REM *             *
18 REM * 99 Magazine *
19 REM *             *
20 REM * G.  Goument *
21 REM ***************
22 G1=48
23 G2=48
24 CALL CLEAR
25 RANDOMIZE
26 DIM A(4)
27 J=0
28 AL=0
29 FOR I=33 TO 47
30 READ A$
```

```
31 CALL CHAR(I,A$)
32 NEXT I
33 DATA 00000000000206
   02,020207,000000000
   0070501,020407,0000
   000000070103,010204
   081020408
34 DATA 00000000000000
   FF,FF00010301010103
   ,FF000000000008,FF
   00030200010203
35 DATA FF008080800000
   8,FF00030001000003,
   FF0080808080808,FF0
   002020203,FF0000008
   0C0808
36 CALL SCREEN(14)
37 CALL COLOR(1,12,1)
38 CALL COLOR(2,12,1)
39 FOR I=9 TO 12
40 CALL COLOR(I,15,1)
41 NEXT I
42 CALL COLOR(3,15,1)
43 CALL CHAR(64,"92543
   8FE385492")
44 CALL CHAR(72,"92543
   8FE385492")
45 CALL CHAR(88,"92543
   8FE385492")
46 CALL CHAR(80,"01010
   7")
47 CALL CHAR(81,"00000
   0000008080A")
48 CALL CHAR(82,"0F020
   2")
49 INPUT "1 ou 2 joueu
   rs  ":N$
50 CALL CLEAR
51 FOR I=1 TO 8
52 CALL COLOR(I,1,1)
53 NEXT I
54 PRINT TAB(9);"Q '''
   ''''''":TAB(8);"%R&@
    @     &":TAB(7);"#P
   &@        &  1":TAB(6
   );"!$&          &"
55 PRINT TAB(6);" &
       &":TAB(7);"()*+
   ,-./":TAB(9);"Q '''
   ''''''":TAB(8);"%R&
            &":TAB(7);"#P
   &         &  2"
56 PRINT TAB(6);"!$&
       @&":TAB(6);" &
      @ @&":TAB(7);"()
   *+,-./":TAB(9);"Q '
   ''''''''":TAB(8);"%R
   &@ @      &"
57 PRINT TAB(7);"#P&@
          &  3":TAB(6);"
   !$&          &":TAB(6)
   ;" &            &":TAB(7
   );"()*+,-./":TAB(9)
   ;"Q '''''''''"
58 PRINT TAB(8);"%R&
          &":TAB(7);"#P&
              &  4":TAB(6)
   ;"!$&           @&":TAB(
   6);" &       @ @&"
59 C=40
60 FOR I=9 TO 16
61 CALL HCHAR(24,I,C)
62 C=C+1
63 NEXT I
64 FOR I=5 TO 23 STEP
   6
65 CALL HCHAR(I,8,34)
66 NEXT I
67 CALL HCHAR(1,3,72)
68 CALL HCHAR(1,30,64)
69 CALL HCHAR(3,3,G1)
70 CALL HCHAR(3,30,G2)
71 CALL COLOR(8,8,1)
72 CALL COLOR(5,16,1)
73 CALL COLOR(6,10,1)
74 CALL COLOR(7,12,1)
75 CALL COLOR(1,12,1)
76 CALL COLOR(2,12,1)
77 IF N$="1" THEN 84
78 DATA 4,17,5,14,5,16
   ,8,13,8,15,9,12,16,
   17,17,14,17,16,20,1
   3,20,15,21,12,0,0
79 RESTORE 78
80 READ Y,X
81 IF Y=0 THEN 84
82 CALL HCHAR(Y,X,72)
83 GOTO 80
84 AL=0
85 RA=0
86 RA=INT((68-1+1)*RND
   )+1
87 N=0
88 Y=0
89 X=0
90 NO=3000
91 IF N$="1" THEN 93
92 IF J=1 THEN 97
93 J=1
94 V=16
95 H=64
96 GOTO 100
97 V=10
98 J=0
99 H=72
100 CALL COLOR(3,2,V)
101 CALL COLOR(4,2,V)
102 CALL KEY(3,T,S)
103 IF N=0 THEN 106
104 IF N>0 THEN 139
105 GOTO 127
106 IF S=0 THEN 102
107 IF T<>49 THEN 111
108 N=1
109 Z=3
110 GOTO 122
111 IF T<>50 THEN 115
112 N=2
113 Z=9
114 GOTO 122
115 IF T<>51 THEN 119
116 N=3
117 Z=15
118 GOTO 122
119 IF T<>52 THEN 124
120 N=4
121 Z=21
122 CALL SOUND(5,123,5
    )
123 GOTO 102
124 IF T<>78 THEN 102
125 IF XA=0 THEN 127
126 CALL HCHAR(YA,XA,H
    )
127 FOR I=1 TO 7
128 CALL SOUND(10,110,
    0)
129 NEXT I
130 Y=0
131 X=0
132 N=0
133 Z=0
134 YA=0
135 XA=0
136 NA=0
137 O=0
138 GOTO 102
139 CALL KEY(3,T,S)
140 IF NA=0 THEN 145
141 IF N=NA+1 THEN 145
142 IF N=NA-1 THEN 145
143 IF N=NA THEN 145
144 GOTO 125
145 CALL HCHAR(Z,22,32
    )
146 IF N<>1 THEN 149
147 CALL HCHAR(Z,22,49
    )
148 GOTO 156
149 IF N<>2 THEN 152
150 CALL HCHAR(Z,22,50
    )
151 GOTO 156
152 IF N<>3 THEN 155
153 CALL HCHAR(Z,22,51
    )
154 GOTO 156
```

```
155 CALL HCHAR(Z,22,52
    )
156 IF S=0 THEN 139
157 IF T<>78 THEN 161
158 N=0
159 CALL SOUND(5,123,1
    0)
160 GOTO 102
161 IF T<>49 THEN 187
162 IF N=1 THEN 171
163 IF N=2 THEN 167
164 IF N=3 THEN 169
165 Y=23
166 GOTO 172
167 Y=11
168 GOTO 172
169 Y=17
170 GOTO 172
171 Y=5
172 CALL SOUND(5,123,5
    )
173 CALL KEY(3,T,S)
174 IF S=0 THEN 173
175 IF T<>49 THEN 178
176 X=10
177 GOTO 264
178 IF T<>50 THEN 181
179 X=12
180 GOTO 264
181 IF T<>51 THEN 184
182 X=14
183 GOTO 264
184 IF T<>52 THEN 172
185 X=16
186 GOTO 264
187 IF T<>50 THEN 213
188 IF N=1 THEN 197
189 IF N=2 THEN 193
190 IF N=3 THEN 195
191 Y=22
192 GOTO 198
193 Y=10
194 GOTO 198
195 Y=16
196 GOTO 198
197 Y=4
198 CALL SOUND(5,123,5
    )
199 CALL KEY(3,T,S)
200 IF S=0 THEN 199
201 IF T<>49 THEN 204
202 X=11
203 GOTO 264
204 IF T<>50 THEN 207
205 X=13
206 GOTO 264
207 IF T<>51 THEN 210
208 X=15
209 GOTO 264
210 IF T<>52 THEN 198
211 X=17
212 GOTO 264
213 IF T<>51 THEN 239
214 IF N=1 THEN 223
215 IF N=2 THEN 219
216 IF N=3 THEN 221
217 Y=21
218 GOTO 224
219 Y=9
220 GOTO 224
221 Y=15
222 GOTO 224
223 Y=3
224 CALL SOUND(5,123,5
    )
225 CALL KEY(3,T,S)
226 IF S=0 THEN 225
227 IF T<>49 THEN 230
228 X=12
229 GOTO 264
230 IF T<>50 THEN 233
231 X=14
232 GOTO 264
233 IF T<>51 THEN 236
234 X=16
235 GOTO 264
236 IF T<>52 THEN 224
237 X=18
238 GOTO 264
239 IF T<>52 THEN 139
240 IF N=1 THEN 249
241 IF N=2 THEN 247
242 IF N=3 THEN 245
243 Y=20
244 GOTO 250
245 Y=14
246 GOTO 250
247 Y=8
248 GOTO 250
249 Y=2
250 CALL SOUND(5,123,5
    )
251 CALL KEY(3,T,S)
252 IF S=0 THEN 251
253 IF T<>49 THEN 256
254 X=13
255 GOTO 264
256 IF T<>50 THEN 259
257 X=15
258 GOTO 264
259 IF T<>51 THEN 262
260 X=17
261 GOTO 264
262 IF T<>52 THEN 250
263 X=19
264 CALL SOUND(5,123,5
    )
265 IF O=1 THEN 280
266 CALL GCHAR(Y,X,E)
267 IF J=0 THEN 270
268 IF E=72 THEN 127
269 GOTO 271
270 IF E=64 THEN 127
271 IF E=32 THEN 127
272 CALL HCHAR(Y,X,88)
273 YA=Y
274 XA=X
275 NA=N
276 O=1
277 Y=0
278 X=0
279 GOTO 102
280 IF N<>NA THEN 291
281 IF Y<>YA THEN 285
282 IF X=XA+2 THEN 298
283 IF X=XA-2 THEN 298
284 GOTO 125
285 IF Y<>YA-1 THEN 28
    8
286 IF X=XA+1 THEN 298
287 GOTO 125
288 IF Y<>YA+1 THEN 12
    5
289 IF X=XA-1 THEN 298
290 GOTO 125
291 IF X<>XA THEN 125
292 IF Y=YA-6 THEN 295
293 IF Y=YA+6 THEN 295
294 GOTO 125
295 CALL GCHAR(Y,X,F)
296 IF F<>32 THEN 125
297 GOTO 307
298 CALL GCHAR(Y,X,F)
299 IF F=32 THEN 307
300 IF J=0 THEN 304
301 CALL HCHAR(Y,X,64)
302 CALL HCHAR(YA,XA,7
    2)
303 GOTO 313
304 CALL HCHAR(Y,X,72)
305 CALL HCHAR(YA,XA,6
    4)
306 GOTO 313
307 IF J=0 THEN 311
308 CALL HCHAR(Y,X,64)
309 CALL HCHAR(YA,XA,3
    2)
310 GOTO 313
311 CALL HCHAR(Y,X,72)
312 CALL HCHAR(YA,XA,3
    2)
313 CALL SOUND(10,200,
    10)
314 CALL SOUND(10,220,
    10)
315 CALL SOUND(10,240,
    10)
```

```
316 CALL SOUND(10,260,
    10)
317 O=0
318 Z=0
319 XA=0
320 YA=0
321 NA=0
322 IF RA>64 THEN 531
323 IF RA>60 THEN 452
324 IF RA>56 THEN 422
325 IF RA>40 THEN 616
326 IF RA>=30 THEN 561
327 IF RA>=20 THEN 482
328 IF RA>=10 THEN 373
329 IF RA=1 THEN 369
330 IF RA=2 THEN 341
331 IF RA=3 THEN 345
332 IF RA=4 THEN 349
333 IF RA=5 THEN 353
334 IF RA=6 THEN 357
335 IF RA=7 THEN 361
336 IF RA=8 THEN 365
337 E=14
338 F=13
339 GOSUB 695
340 GOTO 337
341 E=3
342 F=12
343 GOSUB 695
344 GOTO 341
345 E=4
346 F=11
347 GOSUB 695
348 GOTO 345
349 E=5
350 F=10
351 GOSUB 695
352 GOTO 349
353 E=8
354 F=13
355 GOSUB 695
356 GOTO 353
357 E=9
358 F=12
359 GOSUB 695
360 GOTO 357
361 E=10
362 F=11
363 GOSUB 695
364 GOTO 361
365 E=11
366 F=10
367 GOSUB 695
368 GOTO 365
369 E=2
370 F=13
371 GOSUB 695
372 GOTO 369
373 IF RA=10 THEN 386
374 IF RA=11 THEN 390
375 IF RA=12 THEN 394
376 IF RA=13 THEN 398
377 IF RA=14 THEN 402
378 IF RA=15 THEN 406
379 IF RA=16 THEN 410
380 IF RA=17 THEN 414
381 IF RA=18 THEN 418
382 E=2
383 F=17
384 GOSUB 705
385 GOTO 382
386 E=15
387 F=12
388 GOSUB 695
389 GOTO 386
390 E=16
391 F=11
392 GOSUB 695
393 GOTO 390
394 E=17
395 F=10
396 GOSUB 695
397 GOTO 394
398 E=20
399 F=13
400 GOSUB 695
401 GOTO 398
402 E=21
403 F=12
404 GOSUB 695
405 GOTO 402
406 E=22
407 F=11
408 GOSUB 695
409 GOTO 406
410 E=23
411 F=10
412 GOSUB 695
413 GOTO 410
414 E=2
415 F=13
416 GOSUB 705
417 GOTO 414
418 E=2
419 F=15
420 GOSUB 705
421 GOTO 418
422 IF RA=57 THEN 429
423 IF RA=58 THEN 433
424 IF RA=59 THEN 437
425 E=2
426 F=19
427 GOSUB 441
428 GOTO 425
429 E=2
430 F=13
431 GOSUB 441
432 GOTO 429
433 E=2
434 F=15
435 GOSUB 441
436 GOTO 433
437 E=2
438 F=17
439 GOSUB 441
440 GOTO 437
441 FOR I=1 TO 4
442 IF AL=0 THEN 446
443 CALL HCHAR(E,F,88)
444 GOSUB 738
445 GOTO 447
446 CALL GCHAR(E,F,A(I
    ))
447 E=E+7
448 F=F-1
449 NEXT I
450 IF AL=1 THEN 777
451 GOTO 752
452 IF RA=61 THEN 459
453 IF RA=62 THEN 463
454 IF RA=63 THEN 467
455 E=5
456 F=10
457 GOSUB 471
458 GOTO 455
459 E=5
460 F=16
461 GOSUB 471
462 GOTO 459
463 E=5
464 F=14
465 GOSUB 471
466 GOTO 463
467 E=5
468 F=12
469 GOSUB 471
470 GOTO 467
471 FOR I=1 TO 4
472 IF AL=0 THEN 476
473 CALL HCHAR(E,F,88)
474 GOSUB 738
475 GOTO 477
476 CALL GCHAR(E,F,A(I
    ))
477 E=E+5
478 F=F+1
479 NEXT I
480 IF AL=1 THEN 777
481 GOTO 752
482 IF RA=27 THEN 523
483 IF RA=26 THEN 519
484 IF RA=25 THEN 515
485 IF RA=24 THEN 511
486 IF RA=23 THEN 507
487 IF RA=22 THEN 503
488 IF RA=21 THEN 499
489 IF RA=20 THEN 495
```

```
490 IF RA=28 THEN 527
491 E=20
492 F=13
493 GOSUB 705
494 GOTO 491
495 E=2
496 F=19
497 GOSUB 705
498 GOTO 495
499 E=8
500 F=13
501 GOSUB 705
502 GOTO 499
503 E=8
504 F=15
505 GOSUB 705
506 GOTO 503
507 E=8
508 F=17
509 GOSUB 705
510 GOTO 507
511 E=8
512 F=19
513 GOSUB 705
514 GOTO 511
515 E=14
516 F=13
517 GOSUB 705
518 GOTO 515
519 E=14
520 F=15
521 GOSUB 705
522 GOTO 519
523 E=14
524 F=17
525 GOSUB 705
526 GOTO 523
527 E=14
528 F=19
529 GOSUB 705
530 GOTO 527
531 IF RA=65 THEN 538
532 IF RA=66 THEN 542
533 IF RA=67 THEN 546
534 E=2
535 F=13
536 GOSUB 550
537 GOTO 534
538 E=3
539 F=12
540 GOSUB 550
541 GOTO 538
542 E=4
543 F=11
544 GOSUB 550
545 GOTO 542
546 E=5
547 F=10
548 GOSUB 550
549 GOTO 546
550 FOR I=1 TO 4
551 IF AL=0 THEN 555
552 CALL HCHAR(E,F,88)
553 GOSUB 738
554 GOTO 556
555 CALL GCHAR(E,F,A(I
    ))
556 E=E+6
557 F=F+2
558 NEXT I
559 IF AL=1 THEN 777
560 GOTO 752
561 IF RA=38 THEN 608
562 IF RA=38 THEN 608
563 IF RA=37 THEN 604
564 IF RA=36 THEN 600
565 IF RA=35 THEN 596
566 IF RA=34 THEN 592
567 IF RA=33 THEN 588
568 IF RA=32 THEN 584
569 IF RA=31 THEN 580
570 IF RA=30 THEN 576
571 IF RA=39 THEN 612
572 E=20
573 F=13
574 GOSUB 727
575 GOTO 572
576 E=20
577 F=15
578 GOSUB 705
579 GOTO 576
580 E=20
581 F=17
582 GOSUB 705
583 GOTO 580
584 E=20
585 F=19
586 GOSUB 705
587 GOTO 584
588 E=2
589 F=13
590 GOSUB 727
591 GOTO 588
592 E=2
593 F=19
594 GOSUB 716
595 GOTO 592
596 E=8
597 F=13
598 GOSUB 727
599 GOTO 596
600 E=8
601 F=19
602 GOSUB 716
603 GOTO 600
604 E=14
605 F=13
606 GOSUB 727
607 GOTO 604
608 E=14
609 F=19
610 GOSUB 716
611 GOTO 608
612 E=20
613 F=19
614 GOSUB 716
615 GOTO 612
616 IF RA=41 THEN 635
617 IF RA=42 THEN 639
618 IF RA=43 THEN 643
619 IF RA=44 THEN 647
620 IF RA=45 THEN 651
621 IF RA=46 THEN 655
622 IF RA=47 THEN 659
623 IF RA=48 THEN 663
624 IF RA=49 THEN 667
625 IF RA=50 THEN 671
626 IF RA=51 THEN 675
627 IF RA=52 THEN 679
628 IF RA=53 THEN 683
629 IF RA=54 THEN 687
630 IF RA=55 THEN 691
631 E=5
632 F=19
633 GOSUB 743
634 GOTO 631
635 E=2
636 F=13
637 GOSUB 743
638 GOTO 635
639 E=2
640 F=15
641 GOSUB 743
642 GOTO 639
643 E=2
644 F=17
645 GOSUB 743
646 GOTO 643
647 E=2
648 F=19
649 GOSUB 743
650 GOTO 647
651 E=3
652 F=13
653 GOSUB 743
654 GOTO 651
655 E=3
656 F=15
657 GOSUB 743
658 GOTO 655
659 E=3
660 F=17
661 GOSUB 743
662 GOTO 659
663 E=3
664 F=19
665 GOSUB 743
```

```
666 GOTO 663
667 E=4
668 F=13
669 GOSUB 743
670 GOTO 667
671 E=4
672 F=15
673 GOSUB 743
674 GOTO 671
675 E=4
676 F=17
677 GOSUB 743
678 GOTO 675
679 E=4
680 F=19
681 GOSUB 743
682 GOTO 679
683 E=5
684 F=13
685 GOSUB 743
686 GOTO 683
687 E=5
688 F=15
689 GOSUB 743
690 GOTO 687
691 E=5
692 F=17
693 GOSUB 743
694 GOTO 691
695 FOR I=1 TO 4
696 IF AL=0 THEN 700
697 CALL HCHAR(E,F,88)
698 GOSUB 738
699 GOTO 701
700 CALL GCHAR(E,F,A(I
    ))
701 F=F+2
702 NEXT I
703 IF AL=1 THEN 777
704 GOTO 752
705 FOR I=1 TO 4
706 IF AL=0 THEN 710
707 CALL HCHAR(E,F,88)
708 GOSUB 738
709 GOTO 711
710 CALL GCHAR(E,F,A(I
    ))
711 E=E+1
712 F=F-1
713 NEXT I
714 IF AL=1 THEN 777
715 GOTO 752
716 FOR I=1 TO 4
717 IF AL=0 THEN 721
718 CALL HCHAR(E,F,88)
719 GOSUB 738
720 GOTO 722
721 CALL GCHAR(E,F,A(I
    ))
722 E=E+1
723 F=F-2
724 NEXT I
725 IF AL=1 THEN 777
726 GOTO 752
727 FOR I=1 TO 4
728 IF AL=0 THEN 732
729 CALL HCHAR(E,F,88)
730 GOSUB 738
731 GOTO 733
732 CALL GCHAR(E,F,A(I
    ))
733 E=E+1
734 F=F+1
735 NEXT I
736 IF AL=1 THEN 777
737 GOTO 752
738 FOR P=0 TO 30 STEP
     2
739 CALL SOUND(1,NO,P)
740 NEXT P
741 NO=NO+200
742 RETURN
743 FOR I=1 TO 4
744 IF AL=0 THEN 748
745 CALL HCHAR(E,F,88)
746 GOSUB 738
747 GOTO 749
748 CALL GCHAR(E,F,A(I
    ))
749 E=E+6
750 NEXT I
751 IF AL=1 THEN 777
752 IF A(1)<>H THEN 75
    4
753 CALL SOUND(10,3000
    ,0)
754 IF A(2)<>H THEN 75
    6
755 CALL SOUND(10,3200
    ,0)
756 IF A(3)<>H THEN 75
    8
757 CALL SOUND(10,3400
    ,0)
758 IF A(4)<>H THEN 76
    0
759 CALL SOUND(10,3600
    ,0)
760 E=0
761 F=0
762 IF (A(1)=H)*(A(2)=
    H)*(A(3)=H)*(A(4)=
    H)THEN 764
763 GOTO 87
764 IF J=1 THEN 771
765 G1=G1+1
766 IF G1>57 THEN 808
767 CALL HCHAR(3,3,G1)
768 AL=1
769 CALL SCREEN(16)
770 RETURN
771 AL=1
772 G2=G2+1
773 IF G2>57 THEN 808
774 CALL HCHAR(3,30,G2
    )
775 CALL SCREEN(10)
776 RETURN
777 M$="nouvelle"
778 Y=12
779 X=22
780 GOSUB 790
781 M$="partie"
782 Y=14
783 X=23
784 GOSUB 790
785 M$="o  n"
786 Y=16
787 X=24
788 GOSUB 790
789 GOTO 797
790 FOR I=9 TO 12
791 CALL COLOR(I,2,1)
792 NEXT I
793 FOR I=1 TO LEN(M$)
794 CALL HCHAR(Y,X+I,A
    SC(SEG$(M$,I,1)))
795 NEXT I
796 RETURN
797 CALL KEY(3,T,S)
798 IF S=0 THEN 797
799 IF T<>79 THEN 803
800 CALL SCREEN(14)
801 CALL CLEAR
802 GOTO 50
803 CALL CLEAR
804 CALL SCREEN(13)
805 FOR I=1 TO 12
806 CALL COLOR(I,12,1)
807 NEXT I
808 PRINT TAB(8);"resu
    ltat final":::::
809 IF G1<>G2 THEN 812
810 PRINT TAB(10);"mat
    ch nul"::::::
811 GOTO 816
812 IF G1>G2 THEN 815
813 PRINT "etoiles bla
    nches gagnantes"::
    ::::
814 GOTO 816
815 PRINT "  etoiles r
    ouges gagnantes"::
    ::::
816 FOR DE=1 TO 200
817 NEXT DE
818 END
```

Gestion de fichiers en assembleur et Basic étendu

Michel Teyssou

Nous vous proposons une série de programmes destinés à gérer un fichier d'adresses, et utilisables à partir du Basic étendu, de l'éditeur/assembleur, ou de la Mini-mémoire. Leur mise en œuvre nécessite une imprimante, une extension 32Ko et, bien sûr, un des modules concernés.

Ces programmes offrent les mêmes possibilités que le module "PRK" ou la disquette éditée par Texas Instruments, mais en beaucoup plus performants. En effet, ils autorisent le traitement de 450 fiches, alors que le module "PRK" ne permet de traiter que 102 fiches équivalentes. On peut donc entrer 450 fiches sur une disquette qui contient déjà la totalité des programmes. De plus, la dimension des informations est plus importante qu'avec le module "PRK".

Les enregistrements sont limités à 127 octets, car cela permet, dans la majorité des cas, de loger les informations dont nous avons besoin. De plus, le "Buffer" du PAB emmagasinant à chaque fois 255 octets, cela permet d'avoir en mémoire l'enregistrement demandé plus le suivant, d'où (suivant le type de traitement) un gain de temps appréciable pendant la saisie.

Les saisies sont modifiables ; à cet effet, elles sont regroupées sous forme de "DATA" dans le programme Basic et les zones à modifier sont indiquées dans la version assembleur. En Basic, chaque saisie de chaîne occupe, en format INTERNAL, sa longueur plus un (octet destiné à renseigner l'ordinateur sur la longueur de la chaîne). En assembleur, les indicateurs ont été supprimés afin d'obtenir plus de place ; de ce fait, un fichier créé avec la version "assembleur" ne sera pas utilisable depuis la version "Basic".

La somme maximum des saisies ne doit en aucun cas dépasser la longueur maximum désignée en FIXED (127 dans le cas présent). L'enregistrement 0 contient les renseignements indispensables :

- TRI (nombre de fiches triées) ;
- TOT (totalité des fiches triées et non triées) ;
- Noms des groupes (8 maxi, 11 lettres au plus chacun) ;
- NG (nombre de groupes, ceci avec quelques variantes d'emplacement en Basic et en assembleur).

Les fiches doivent être regroupées en catégories : secteurs ou groupes (nous avons choisi ce dernier qualificatif, mais on peut prendre n'importe lequel). Si la fiche n'est pas dans un des groupes, elle sera classée "INACTIF" et pourra être effacée lors d'un tri.

Grâce à un tri par indexation, on peut classer très rapidement les noms par ordre alphabétique. Pour donner un ordre d'idée, il faut 45 minutes au module "PRK" pour trier 200 noms, alors qu'il faut moins de cinq minutes au programme "TRI" écrit en Basic et environ trois fois moins à celui écrit en assembleur.

Il faudra se servir souvent de ces programmes "TRI", car lors de la saisie des noms, la recherche pour vérifier qu'il n'est pas déjà dans le fichier sera extrêmement rapide (méthode des blocs). Sinon, la recherche se fait séquentiellement, dont beaucoup plus lentement.

Dans tous les programmes d'impression ou d'affichage, on peut à tout moment interrompre le processus avec la barre "espace" (idem pour reprendre). On peut arrêter avec <FCTN BACK>. Dans ce cas, il faut maintenir les 2 touches enfoncées jusqu'à l'arrêt.
Dans la totalité des saisies de variables numériques, le fait de rentrer le chiffre zéro (0), permet un retour au dernier menu, et invalide la saisie en cours.

Programmes Basic

Conseil de saisie

Lorsque vous rentrerez ces programmes dans votre ordinateur préféré, évitez de mettre les remarques et tête de chapitre ; elles ne sont là que pour clarifier le listing et aucun branchement ne se fait sur ces lignes. De ce fait, votre programme se manipulera plus rapidement au chargement, et tiendra moins de place sur la disquette, (les programmes + "données" et "index " occupent la totalité de la face disquette avec les 450 fiches à condition que les programmes n'aient aucun commentaire). J'ai cherché à les

optimiser au mieux, et du fait de l'imprimante, ils tournent pratiquement aussi vite (pour l'impression) que des programmes similaires écrits en assembleur.

Sommaire (LOAD)

Programme qui permet de choisir le traitement désiré ; il n'y a rien de spécial à dire sur son fonctionnement qui est hyper classique. Le fait qu'il s'appelle LOAD permet de le charger automatiquement.

Programme "LOAD"

Basic étendu

```
10 !***********************
20 !*   SOMMAIRE
30 !***********************
40 CALL CLEAR :: CALL COLOR(13,8,1
   ):: CALL VCHAR(1,31,129,96):: C
   ALL SCREEN(8):: FOR I=1 TO 12 :
   : CALL COLOR(I,16,5):: NEXT I
50 DISPLAY AT(4,8)BEEP:"SOMMAIRE":
   TAB(8);"~~~~~~~~"
60 DISPLAY AT(8,6):"1 GERER": :TAB
   (6);"2 TRIER": :TAB(6);"3 IMPRI
   MER": :TAB(6);"4 INITALISER": :
   TAB(6);"5 RECHERCHER"
70 DISPLAY AT(18,6):"6 TITRER EN G
   ROS"
80 CALL KEY(0,K,S):: IF S=0 THEN 8
   0
90 IF K<49 OR K>54 THEN 80
100 ON K-48 GOTO 110,120,130,140,1
    50,160
110 CALL HCHAR(8,7,62):: RUN "DSK1
    .GERER"
120 CALL HCHAR(10,7,62):: RUN "DSK
    1.TRIER"
130 CALL HCHAR(12,7,62):: RUN "DSK
    1.IMPRIMER"
140 CALL HCHAR(14,7,62):: RUN "DSK
    1.INITIALISE"
150 CALL HCHAR(16,7,62):: RUN "DSK
    1.RECHERCHER"
160 CALL HCHAR(18,7,62):: RUN "DSK
    1.TITRER"
170 END
```

Initialise le fichier

C'est le premier programme à faire tourner ; il permet d'ouvrir les fichiers, de contrôler si l'espace initial pour les données est suffisant. Vous devrez alors rentrer le nom des groupes dans lesquels vous mettrez vos fiches.

Programme "INITIALISE"

Basic étendu

```
10 !***********************
20 !INITIALISER LE FICHIER
30 !***********************
40 ON WARNING NEXT :: CALL CLEAR :
   : CALL SCREEN(7):: CALL COLOR(
   13,7,7):: CALL VCHAR(1,31,129,
   96):: ON BREAK NEXT
50 FOR A=1 TO 12 :: CALL COLOR(A,5
   ,12):: NEXT A :: CALL TIT :: C
   ALL ENT :: CALL STOP(A)
60 OPEN #1:"DSK1.",INPUT ,INTERNAL
   ,RELATIVE
70 INPUT #1:A$,A,B,C :: CLOSE #1
80 IF C>227 THEN 140
90 CALL NET :: DISPLAY AT(2,1):"LE
    NOMBRE DE SECTEURS SUR LA":"D
   ISQUETTE EST INSUFFISANT...":"
   -----------------------------"
100 DISPLAY AT(5,1):" LE FICHIER C
    OMPLET OCCUPE   227 SECTEURS P
    OUR 450 NOMS -----------------
    -----------"
110 DISPLAY AT(10,2):"NOM DU DISQU
    E:";" ";A$: :" SECTEURS DISP:"
    ;C
120 DISPLAY AT(15,7):"OPTIONS:": :
    TAB(6);"1.AUTRE ESSAI": :TAB(6
    );"2.SORTIE"
130 CALL CHOIX(A):: IF A=1 THEN 60
     ELSE STOP
140 CALL NET :: DISPLAY AT(5,1):"-
    -----------------------------":"
     LE FICHIER COMPLET OCCUPERA 2
    27 SECTEURS POUR 450 NOMS ----
    --------------------------"
150 DISPLAY AT(15,5):"OUVERTURE DU
     FICHIER"
160 CALL ENT :: CALL STOP(A)
170 OPEN #1:"DSK1.DONNEES",RELATIV
    E,INTERNAL,FIXED 127
180 !***********************
190 !*  Nom de groupes
200 !***********************
210 CALL NET
220 DIM G$(8)
230 CALL GROUP :: FOR A=1 TO 8 ::
    DISPLAY AT(9+A,2):A;"- ";G$(A)
    :: NEXT A
240 NG,A=0
250 CALL NGR :: CALL NUM(A)
260 IF A>8 THEN 340
270 IF G$(A)<>"" AND G$(A)<>" " TH
    EN IF NG THEN NG=NG-1
280 ACCEPT AT(23,5)SIZE(-11):G$(A)
290 IF G$(A)="" OR G$(A)=" " THEN
    310
300 NG=NG+1 :: IF NG=9 THEN CALL N
    ET2 :: DISPLAY AT(9+A,7):G$(A)
    :: CALL UTIL :: CALL ENT :: CA
    LL STOP(A):: GOTO 340
310 DISPLAY AT(9+A,7):G$(A)
320 DISPLAY AT(23,2):"  ENCORE (O/
    N)? O" :: ACCEPT AT(23,18)VALI
    DATE("ON")SIZE(-1):A$
330 IF A$="O" THEN CALL NET2 :: GO
    TO 250
340 IF NG=0 THEN DISPLAY AT(21,2):
    "AUCUN No DE GROUPE VALIDE" ::
     CALL ENT :: CALL STOP(A):: CA
    LL NET2 :: GOTO 240
350 CALL NET2 :: DISPLAY AT(21,2):
    "MODIFICATIONS (O/N)?N" :: ACC
    EPT AT(21,22)SIZE(-1)VALIDATE(
    "ON"):A$
360 IF A$="O" THEN CALL NET2 :: GO
    TO 240
370 CALL NET :: DISPLAY AT(14,3):"
    DISPOSITIF D'IMPRESSION ?": :T
    AB(6);"-->" :: ACCEPT AT(16,10
    )BEEP SIZE(-16):IMP$
380 DISPLAY AT(14,2)BEEP:"SI TOUT
    EST OK APPUYEZ SUR": :"     <EN
    TER> POUR VALIDER": :"     <BAC
    K> POUR CORRIGER"
390 CALL STOP(A):: IF A=15 THEN CA
    LL NET :: GOTO 230
400 TRI$="350" :: TOT$="000" :: TI
    TRE$="O" :: NG$=STR$(NG)
410 PRINT #1,REC 0:TRI$,TOT$,TITRE
    $,IMP$,G$(1),G$(2),G$(3),G$(4)
    ,G$(5),G$(6),G$(7),G$(8),NG$ :
    : CLOSE #1
420 OPEN #3:"DSK1.INDEX",INTERNAL,
    OUTPUT,FIXED 4
430 PRINT #3:" " :: CLOSE #3 :: CA
    LL NET :: DISPLAY AT(9,4)BEEP:
    "**********************":TAB(4
    );"*                    *"
440 DISPLAY AT(11,4)BEEP:"* FICHIE
    R INITIALISE *":TAB(4);"*
                        *":TAB(4);"****
    ******************"
450 CALL ENT :: CALL STOP(A)
460 END
470 !------------------
480 ! S/PROGRAMMES FERMES
490 !------------------
500 SUB TIT :: DISPLAY AT(9,10):"T
    I 99/4A": : :TAB(4);"INITIALIS
    ER LE FICHIER":TAB(4);"~~~~~~~
    ~~~~~~~~~~~~~~" :: SUBEND
510 SUB ENT :: DISPLAY AT(23,1):"
        Appuyez sur <enter>" :: SU
    BEND
520 SUB GROUP :: DISPLAY AT(5,5):"
    GROUPES SUR FICHIER":TAB(5);"~
    ~~~~~~~~~~~~~~~~~~" :: SUBEND
530 SUB NGR :: DISPLAY AT(21,4)BEE
    P:"NOM DU GROUPE No" :: SUBEND
540 SUB NUM(G):: G=G+1 :: DISPLAY
    AT(21,20):G :: SUBEND
550 SUB UTIL :: DISPLAY AT(21,2):"
    LES 8 No DE GROUPES ONT       E
    TES UTILISES" :: SUBEND
560 SUB STOP(K)
570 CALL SOUND(150,800,0)
580 CALL KEY(0,K,S):: IF NOT S THE
    N 580
590 IF K<>13 AND K<>15 THEN 570
600 SUBEND
610 SUB NET :: CALL VCHAR(1,3,32,6
    72):: SUBEND
620 SUB NET2 :: FOR A=20 TO 24 ::
    CALL HCHAR(A,3,32,28):: NEXT A
     :: SUBEND
```

Gestion du fichier

Ce programme permet de saisir les données et de modifier éventuellement noms de groupes ou informations contenues dans les fiches. Il permet également de marquer une fiche pour l'effacer (avec TRI), de rechercher une fiche ou un goupe et de l'imprimer sur l'écran. Les lignes 1210 à 1230 contiennent les 7 groupes d'informations de la fiche (3 à 9), le nombre à droite correspond au nombre de lettres maxi que contiendra la saisie. En cas de modifications, ce total ne doit en aucun cas dépasser 109 pour les 7 groupes de saisies.

En effet, chaque enregistrement contient 9 informations D$ (1à9), la première occupe 1 octet, soit A (actif), I (inactif), E (effacement). La deuxième occupe 8 octets (groupe dans lequel est la personne), les autres sont ceux de la fiche, soit 1+8+109 = 118+1 octet par information : 18 + 9 = 127.

Programme "GERER"

Basic étendu

```
10 !***********************
11 !* GESTION DU FICHIER
12 !***********************
20 DIM G$(8),D$(9),XX(9),P$(450)
30 ON WARNING NEXT :: ON BREAK NEX
   T :: CALL CLEAR :: CALL SCREEN(
   7):: CALL COLOR(13,7,7):: CALL
```

```
   VCHAR(1,31,129,96)
40 FOR A=1 TO 12 :: CALL COLOR(A,5
   ,12):: NEXT A :: CALL TIT :: CA
   LL ENT :: CALL STOP(RG$,A)
50 DISPLAY AT(23,5):"Un instant sv
   p...."
60 OPEN #1:"DSK1.",RELATIVE,INTERN
   AL,INPUT
70 INPUT #1:A$ :: IF A$="DONNEES"
   THEN 100 ELSE IF A$<>"" THEN 70
80 CLOSE #1 :: CALL NET :: CALL RI
   EN
90 CALL CHOIX(A):: IF A<1 OR A>2 T
   HEN 90 :: IF A=1 THEN 50 ELSE S
   TOP
100 CLOSE #1
110 OPEN #1:"DSK1.DONNEES",RELATIV
    E,INTERNAL,FIXED 127
120 INPUT #1,REC 0:TRI$,TOT$,TITRE
    $,IMP$,G$(1),G$(2),G$(3),G$(4)
    ,G$(5),G$(6),G$(7),G$(8),NG$ :
    : TRI=VAL(TRI$):: TOT=VAL(TOT$
    ):: NG=VAL(NG$)
130 OPEN #3:"DSK1.INDEX",INTERNAL,
    INPUT ,FIXED 4
140 FOR I=1 TO TRI :: INPUT #3:P$(
    I):: NEXT I :: CLOSE #3
150 GROUP$="NNNNNNNN" :: RG$="N"
160 CALL NET :: RESTORE #1 :: CALL
    OPT1
170 DISPLAY AT(17,5):"FICHES TRIEE
    S =";TRI: :TAB(8);"NON TRIEES
    =";TOT-TRI
180 CALL CHOIX(A):: IF A<1 OR A>4
    THEN 180
190 ON A GOTO 210,720,990,1200
195 !**********************
200 !Mise a jour du fichier
205 !**********************
210 CALL NET :: CALL OPT2
220 CALL CHOIX(A):: IF A<1 OR A>4
    THEN 220
230 IF (A=1)*(TOT>450)=0 THEN 250
240 CALL NET :: CALL PLEIN :: CALL
     ENT :: CALL STOP(RG$,A):: GOT
    O 210
250 ON A GOTO 270,500,680,160
260 !***** Ajouter ******
270 CALL NET :: CALL NOM :: CALL A
    JOUT :: GOSUB 1450 :: IF N$="
    " OR N$="" THEN
480 ELSE IF A=0 THEN 340
280 CALL DEJA :: IF D$(1)="E" THEN
     CALL EFF ELSE IF D$(1)="I" TH
    EN CALL INAC :: CALL AUTRE ELS
    E CALL AUTRE
290 ACCEPT AT(24,24)VALIDATE("ON")
    SIZE(-1):A$ :: IF A$="N" THEN
    270
300 REP=Z :: IF D$(1)<>"E" THEN 34
    0
310 IF D$(2)=GROUP$ THEN D$(1)="I"
    GOTO 460
320 D$(1)="A" :: GOTO 460
330 RESTORE 1220 :: CALL NET :: CA
    LL DONNE :: CALL MOD
340 J=409 :: GOSUB 1400 :: D$(3)=N
    $
350 DISPLAY AT(24,2)BEEP:"C'EST CO
    RRECT? (O/N) O"
360 ACCEPT AT(24,23)VALIDATE("ON")
    SIZE(-1):A$ :: IF A$="N" THEN
    RESTORE 1220 :: DISPLAY AT(23,
    1):: GOTO 340
370 D$(2)=GROUP$ :: IF NG=0 THEN D
    $(1)="I" :: GOTO 450
380 CALL NET :: CALL NOU :: DISPLA
    Y AT(9,1):"--------------------
    ---------":TAB(2);D$(8);" ";D
    $(3):"-------------------------
    ----"
390 FOR A=1 TO NG
400 DISPLAY AT(12+A,5):G$(A);TAB(1
    7);"O/N N" :: ACCEPT AT(12+A,2
    1)VALIDATE("ON")SIZE(-1)BEEP:A
    $
410 GOSUB 1510
420 NEXT A
430 DISPLAY AT(24,3):"C'EST CORREC
    T? (O/N) O" :: ACCEPT AT(24,24
    )VALIDATE("ON")SIZE(-1):A$ ::
    IF A$="N" THEN D$(2)=GROUP$ ::
    GOTO 380
440 IF D$(2)=GROUP$ THEN D$(1)="I"
    ELSE D$(1)="A"
450 IF REP=0 THEN Z=TOT+1 ELSE Z=R
    EP
460 GOSUB 1280 :: IF REP=0 THEN TO
    T=TOT+1
470 GOSUB 1250
480 CALL CONT :: ACCEPT AT(24,24)V
    ALIDATE("ON")SIZE(-1):A$ :: IF
    A$="N" THEN 160 ELSE 270
490 !***** Modifier *****
500 CALL NET :: CALL NOM :: CALL M
    OD :: GOSUB 1450
510 RG$="N" :: IF A=0 THEN CALL NE
    T :: CALL PAS :: CALL ENT :: C
    ALL STOP(RG$,A):: GOTO 160
520 CALL NET :: RESTORE 1210
530 FOR A=3 TO 9 :: READ A$,XX(A)
540 DISPLAY AT(2+ZZ,1):A;"- ";SEG$
    (A$,1,POS(A$,".",1)-1):D$(A)
550 ZZ=ZZ+3 :: NEXT A
560 ZZ=0 :: CALL LIGNE
570 ACCEPT AT(24,23)VALIDATE(DIGIT
    )SIZE(1):B :: IF B=0 THEN 660
    ELSE IF B=1 THEN 590
580 CALL DON :: ACCEPT AT(24,2)SIZ
    E(-XX(B))BEEP:A$ :: D$(B)=A$ :
    : GOTO 520
590 CALL NET :: W=1 :: CALL GR ::
    GOSUB 1480 :: CALL CH2
600 ACCEPT AT(24,19)VALIDATE("0123
    ")SIZE(1)BEEP:B :: IF B=0 THEN
     660 ELSE IF B=1 THEN 650
610 IF B=2 THEN CALL AJN ELSE CALL
     EFG
620 ACCEPT AT(24,26)SIZE(1)VALIDAT
    E("12345678"):A
630 IF B=2 THEN A$="O" ELSE A$="N"
     :: GOSUB 1510 :: GOTO 590
640 GOSUB 1510 :: GOTO 590
650 GOSUB 1280
660 CALL MISE :: ACCEPT AT(24,28)S
    IZE(-1)VALIDATE("ON")BEEP:A$ :
    : IF A$="N" THEN 160 ELSE 210
670 !****** Effacer ******
680 CALL NET :: CALL NOM :: CALL E
    FFA :: GOSUB 1450 :: IF A=0 TH
    EN CALL NET :: CALL PAS :: CAL
    L ENT :: CALL STOP(RG$,A):: GO
    TO 160
690 D$(1)="E" :: GOSUB 1280 :: CAL
    L AEFF :: ACCEPT AT(24,27)VALI
    DATE("ON")SIZE(-1):A$
700 IF A$="O" THEN 680 ELSE 160
705 !**********************
710 !* Nom de groupes
715 !**********************
720 CALL NET :: CALL OPT4 :: CALL
    CHOIX(A)
730 IF A<1 OR A>3 THEN 730
740 ON A GOTO 760,910,160
750 !** Ajouter un groupe **
760 IF NG=8 THEN CALL UTIL :: CALL
     ENT :: CALL STOP(RG$,A):: GOT
    O 160
770 CALL NET :: W=0 :: CALL GROUP
    :: GOSUB 1480 :: CALL NOUV
780 ACCEPT AT(22,26)VALIDATE(DIGIT
    )SIZE(1)BEEP:A :: IF A=0 THEN
    720
790 IF A<1 OR A>8 THEN 780
800 Y=1 :: IF G$(A)="" OR G$(A)="
    " THEN 840
810 CALL REMP :: ACCEPT AT(24,27)V
    ALIDATE("ON")SIZE(-1):A$
820 IF A$="N" THEN CALL CONT :: GO
    TO 890
830 Y=0 :: GOTO 850
840 NG=NG+1
850 CALL NGR :: DISPLAY AT(24,1)::
     ACCEPT AT(24,2)SIZE(-11):G$(A
    )
860 IF (G$(A)="")+(G$(A)=" ")THEN
    NG=NG-1
870 GOSUB 1250 :: IF Y<>1 THEN GOS
    UB 1660
880 CALL NET :: CALL CONT
890 ACCEPT AT(24,24)VALIDATE("ON")
    SIZE(-1):A$ :: IF A$="N" THEN
    160 ELSE 760
900 !** Effacer un groupe **
910 IF NG=0 THEN CALL NET :: CALL
    PASG :: CALL ENT :: CALL STOP(
    RG$,A):: GOTO 720
920 CALL NET :: W=0 :: CALL GROUP
    :: GOSUB 1480 :: CALL EFG :: A
    CCEPT AT(24,24)VALIDATE(DIGIT)
    SIZE(1):A
930 IF A=0 THEN 720 ELSE IF A>8 TH
    EN 920
940 IF G$(A)="" OR G$(A)=" " THEN
    960
950 G$(A)="" :: NG=NG-1 :: GOSUB 1
    250 :: GOSUB 1660 :: CALL NET
960 CALL AEFF :: ACCEPT AT(24,27)V
    ALIDATE("ON")SIZE(-1):A$
970 IF A$="O" THEN 910 ELSE 160
975 !**********************
980 !*  Recherche
985 !**********************
990 CALL NET :: CALL OPT3 :: CALL
    CHOIX(A):: IF A<1 OR A>4 THEN
    990 :: X=0
1000 ON A GOTO 1020,1060,1110,160
1010 !**** Une fiche *****
1020 CALL NET :: CALL NOM :: CALL
     AFF :: GOSUB 1450
1030 IF A=0 THEN CALL NET :: CALL
     PAS :: CALL ENT :: CALL STOP(
     RG$,A):: GOTO 160
1040 GOSUB 1560 :: IF A=13 THEN DI
     SPLAY AT(24,1):: CALL ATR ::
     ACCEPT AT(23,24)VALIDATE("ON"
     )SIZE(-1)BEEP:A$ :: IF A$="O"
      THEN 990 ELSE 160
1050 !**** Un groupe *****
1060 IF NG=0 THEN CALL NET :: CALL
      PAS :: CALL ENT :: CALL STOP
     (RG$,A):: GOTO 160
1070 CALL NET :: W=0 :: CALL GROUP
      :: GOSUB 1480 :: CALL GAF
1080 ACCEPT AT(24,24)VALIDATE(DIGI
     T)SIZE(-1):Y :: IF Y=0 THEN 9
     90 ELSE IF Y>8 THEN 1080
1090 IF G$(Y)="" OR G$(Y)=" " THEN
      1080 ELSE 1120
1100 !* Toutes les fiches *
1110 X=1
1120 FOR Z=1 TO TOT
1130 GOSUB 1270
1140 IF X<>1 THEN IF SEG$(D$(2),Y,
     1)="N" THEN 1160
1150 GOSUB 1560
1160 NEXT Z
1170 CALL NET :: CALL FIN :: CALL
     ENT :: CALL STOP(RG$,A)
1180 IF A=13 THEN CALL ATR :: ACCE
     PT AT(23,24)VALIDATE("ON")SIZ
     E(-1):A$ :: IF A$="O" THEN 99
     0
1190 GOTO 160
1200 CLOSE #1 :: STOP
1201 !-----------------
1202 ! ZONE DES DATAS
1203 !-----------------
1210 DATA NOM PRENOM..............
     ,23
1220 DATA ADRESSE.................
     ....,27,CODE POSTAL.,5,VILLE..
     ..........,15
1230 DATA NO TEL.......,12,TITRE.,
     5,DIVERS..................,22
1240 DATA INACTIF,EFFACEMENT
1242 !-----------------
1243 ! ZONE S/P OUVERTS
1244 !-----------------
1250 TRI$=STR$(TRI):: TOT$=STR$(TO
     T):: NG$=STR$(NG)
1255 PRINT #1,REC 0:TRI$,TOT$,TITR
     E$,IMP$,G$(1),G$(2),G$(3),G$(
     4),G$(5),G$(6),G$(7),G$(8),NG
     $ :: RETURN
1260 INPUT #1,REC Z:D$(1),D$(2),D$
     (3):: RETURN
1270 INPUT #1,REC Z:D$(1),D$(2),D$
     (3),D$(4),D$(5),D$(6),D$(7),D
     $(8),D$(9):: RETURN
1280 PRINT #1,REC Z:D$(1),D$(2),D$
     (3),D$(4),D$(5),D$(6),D$(7),D
```

```
     $(8),D$(9):: RETURN
1290 A=0 :: IF TRI=0 THEN 1360 ELS
     E IF TOT THEN I=1 :: J=TRI EL
     SE RETURN
1300 IF D$(3)="" OR D$(3)=" " THEN
      RETURN
1310 ZT=INT((I+J)/2):: Z=VAL(P$(ZT
     )):: GOSUB 1260
1320 IF D$(3)=N$ THEN A=Z :: GOSUB
      1270 :: RETURN
1330 IF D$(3)<N$ THEN I=ZT+1 ELSE
     J=ZT-1
1340 IF (ZT<1)+(I>J)=0 THEN 1310
1350 IF TRI=TOT THEN RETURN
1360 FOR Z=TRI+1 TO TOT
1370 GOSUB 1260
1380 IF D$(3)=N$ THEN A=Z :: GOSUB
      1270 :: RETURN
1390 NEXT Z :: Z=0 :: RETURN
1400 A=INT(J/100)
1410 FOR A=A TO J-A*100 :: READ A$
     ,I
1420 DISPLAY AT(16,2):A$: :"?" ::
     CALL SOUND(150,400,10)
1430 ACCEPT AT(18,2)SIZE(-I):D$(A)
1440 NEXT A :: RETURN
1450 J=303 :: RESTORE 1210
1460 GOSUB 1400 :: N$=D$(3)
1470 GOSUB 1290 :: RETURN
1480 FOR A=1 TO 8 :: DISPLAY AT(7+
     A,2):A;TAB(6);"- ";G$(A):: IF
      W=0 OR SEG$(D$(2),A,1)="N" T
     HEN 1500
1490 DISPLAY AT(7+A,23):"*"
1500 NEXT A :: RETURN
1510 IF A=1 THEN D$(2)=A$&SEG$(D$(
     2),2,7):: GOTO 1540
1520 IF A=8 THEN D$(2)=SEG$(D$(2),
     1,7)&A$ :: GOTO 1540
1530 D$(2)=SEG$(D$(2),1,A-1)&A$&SE
     G$(D$(2),A+1,8-A)
1540 IF D$(1)="E" THEN RETURN ELSE
      IF D$(2)=GROUP$ THEN D$(1)="
     I" :: RETURN ELSE D$(1)="A" :
     : RETURN
1560 CALL NET :: DISPLAY AT(2,1):D
     $(8):D$(3): :D$(4):D$(5);" ";
     D$(6):D$(7):D$(9)
1570 C=0 :: IF D$(1)="I" THEN 1610
1580 FOR A=1 TO NG :: IF SEG$(D$(2
     ),A,1)="N" THEN 1600
1590 DISPLAY AT(14+C,18):G$(A):: C
     =C+1
1600 NEXT A
1610 IF D$(1)="A" THEN 1650
1620 RESTORE 1240
1630 IF D$(1)="I" THEN READ A$ ELS
     E READ A$ :: READ A$
1640 DISPLAY AT(1,18):A$
1650 CALL BACK :: RG$="O" :: CALL
     STOP(RG$,A):: RG$="N" :: IF A
     =13 THEN RETURN ELSE Z=TOT+1
     :: RETURN
1660 IF TOT THEN CALL NET :: CALL
     MOM :: A$="N" ELSE RETURN
1670 FOR Z=1 TO TOT :: GOSUB 1270
     :: IF D$(1)="E" OR SEG$(D$(2)
     ,A,1)=A$ THEN 1690
1680 GOSUB 1510 :: GOSUB 1280
1690 NEXT Z :: RETURN
1695 !-------------------
1696 ! ZONE S/P FERMES
1697 !-------------------
1700 SUB BACK :: DISPLAY AT(24,7):
     "<ENTER OU BACK>" :: SUBEND
1710 SUB OPT4 :: DISPLAY AT(5,7):"
     NOMS DE GROUPES":TAB(7);"~~~~
     ~~~~~~~~~~~": : :TAB(6);"1.AJ
     OUTER UN NOM"
1720 DISPLAY AT(11,5):TAB(6);"2.EF
     FACER UN NOM": :TAB(6);"3.RET
     OUR" :: SUBEND
1730 SUB TIT :: DISPLAY AT(9,11):"
     TI 99/4A": : :TAB(6);"GESTION
      DU FICHIER":TAB(6);"~~~~~~~~
     ~~~~~~~~~~" :: SUBEND
1740 SUB ENT :: DISPLAY AT(23,5):"
     Appuyez sur <enter>" :: SUBEN
     D
1750 SUB OPT1 :: DISPLAY AT(3,5):"
     OPTIONS DU FICHIER":TAB(5);"~
```

```
     ~~~~~~~~~~~~~~~~~": : :TAB(5)
     ;"1.MISE A JOUR": :TAB(5);"2.
     NOMS DE GROUPES"
1760 DISPLAY AT(11,5):"3.RECHERCHE
     ": :TAB(5);"4.SORTIE" :: SUBE
     ND
1770 SUB CHOIX(A):: DISPLAY AT(23,
     6)BEEP:"VOTRE CHOIX ?" :: ACC
     EPT AT(23,20)SIZE(1):A :: SUB
     END
1780 SUB RIEN :: DISPLAY AT(7,2):"
     LES DONNEES NE SONT PAS": :"
      SUR CETTE DISQUETTE !": : :
     :TAB(9);"OPTIONS":TAB(9);"~~~
     ~~~~"
1790 DISPLAY AT(17,7):"1.AUTRE ESS
     AI": :TAB(7);"2.SORTIE" :: SU
     BEND
1800 SUB OPT3 :: DISPLAY AT(5,8):"
     RECHERCHE":TAB(8);"~~~~~~~~~"
     : : :TAB(7);"1.UNE FICHE": :T
     AB(7);"2.UN GROUPE"
1810 DISPLAY AT(13,7):"3.TOUT LE F
     ICHIER": :TAB(7);"4.RETOUR" :
     : SUBEND
1820 SUB OPT2 :: DISPLAY AT(5,8):"
     MISE A JOUR":TAB(8);"~~~~~~~~
     ~~~": :TAB(8);"1.AJOUTER": :T
     AB(8);"2.MODIFIER"
1830 DISPLAY AT(12,8):"3.EFFACER":
      :TAB(8);"4.RETOUR" :: SUBEND
1840 SUB TITRE :: DISPLAY AT(16,2)
     :"TITRE.": :"?" :: SUBEND
1850 SUB DONNE :: DISPLAY AT(4,2):
     "RENTREZ LES DONNEES": :"QUE
     VOUS VOULEZ" :: SUBEND
1860 SUB NOM :: DISPLAY AT(4,2):"R
     ENTREZ LE NOM DE LA": :" PERS
     ONNE QUE VOUS VOULEZ" :: SUBE
     ND
1870 SUB AJOUT :: DISPLAY AT(8,2):
     "AJOUTER." :: SUBEND
1880 SUB MOD :: DISPLAY AT(8,2):"M
     ODIFIER." :: SUBEND
1890 SUB EFFA :: DISPLAY AT(8,2):"
     EFFACER." :: SUBEND
1900 SUB AFF :: DISPLAY AT(8,2):"A
     FFICHER." :: SUBEND
1910 SUB GROUP :: DISPLAY AT(5,5):
     "GROUPES SUR FICHIER":TAB(5);
     "~~~~~~~~~~~~~~~~~~~" :: SUBE
     ND
1920 SUB NOU :: DISPLAY AT(2,4):"I
     NDIQUEZ PAR (O/N) LE": :" GRO
     UPE DANS LEQUEL METTRE": :"
      LE NOUVEAU NOM." :: SUBEND
1930 SUB CONT :: DISPLAY AT(24,1):
     "  VOUS CONTINUEZ (O/N)?O" ::
      SUBEND
1940 SUB MISE :: DISPLAY AT(24,1):
     "UNE AUTRE MISE A JOUR(O/N)?O
     " :: SUBEND
1950 SUB AEFF :: DISPLAY AT(24,2)B
     EEP:"AUTRE MODIFICATION (O/N)
     ?O" :: SUBEND
1960 SUB PLEIN :: DISPLAY AT(12,2)
     :"DESOLE LE FICHIER EST PLEIN
     " :: SUBEND
1970 SUB PAS :: DISPLAY AT(12,3):"
     CETTE PERSONNE N'EST PAS": :"
        SUR FICHIER" :: SUBEND
1980 SUB DEJA :: DISPLAY AT(22,2):
     "PERSONNE DEJA SUR FICHIER" :
     : SUBEND
1990 SUB AUTRE :: DISPLAY AT(24,2)
     BEEP:"    CHANGEMENT (O/N)? N
     " :: SUBEND
2000 SUB EFF :: DISPLAY AT(23,2)BE
     EP:"ET MARQUEE EFFACEMENT":"
     REINTRODUCTION (O/N)?" :: SUB
     END
2010 SUB INAC :: DISPLAY AT(23,2)B
     EEP:"ET MARQUEE INACTIVE" ::
     SUBEND
2020 SUB GR :: DISPLAY AT(2,1):"
       GROUPE            MEMBRE" ::
     SUBEND
2030 SUB CH2 :: DISPLAY AT(20,1):"
       1 POUR INCHANGE":"  2 POUR
     AJOUTER A UN GROUPE":"  3 POU
     R EFFACER D'UN GROUPE": :"
     VOTRE CHOIX ?" :: SUBEND
```

```
2040 SUB AJN :: DISPLAY AT(24,2):"
        AJOUTER A QUEL NO ?" :: SUB
     END
2050 SUB PASG :: DISPLAY AT(12,2):
     "AUCUN GROUPE SUR FICHIER":"
     ~~~~~~~~~~~~~~~~~~~~~~~~" ::
     SUBEND
2060 SUB NOUV :: DISPLAY AT(22,6):
     "NOUVEAU GROUPE No ?" :: SUBE
     ND
2070 SUB REMP :: DISPLAY AT(24,2):
     "REMPLACEMENT GROUPE O/N? O"
     :: SUBEND
2080 SUB NGR :: DISPLAY AT(22,2):"
     NOM DU NOUVEAU GROUPE ?":"
         " :: SUBEND
2090 SUB UTIL :: DISPLAY AT(23,2):
     "LES 8 GROUPES SONT UTILISES"
      :: SUBEND
2100 SUB EFG :: DISPLAY AT(24,2):"
     NO GROUPE A EFFACER ?" :: SUB
     END
2110 SUB MOM :: DISPLAY AT(12,2):"
     MODIFICATION EN COURS": :" UN
       MOMENT SVP..." :: SUBEND
2120 SUB LIGNE :: DISPLAY AT(23,2)
     BEEP:"NO DE LA LIGNE A CHANGE
     R":" 1 POUR INCHANGE   ?" ::
     SUBEND
2130 SUB DON :: DISPLAY AT(23,2):"
     NOUVELLE DONNEE":"?" :: SUBEN
     D
2140 SUB GAF :: DISPLAY AT(24,1):"
         GROUPE A AFFICHER ?" :: SU
     BEND
2150 SUB FIN :: DISPLAY AT(12,7):"
     FIN DU FICHIER":TAB(7);"~~~~~
     ~~~~~~~~~" :: SUBEND
2160 SUB ATR :: DISPLAY AT(23,2):"
     AUTRE RECHERCHE? (O/N)N" :: S
     UBEND
2170 SUB STOP(RG$,A)
2180 CALL SOUND(150,800,0)
2190 CALL KEY(0,A,B):: IF NOT B TH
     EN 2190 :: IF A<>13 AND(A=15)
     *(RG$="O")=0 THEN 2180
2200 SUBEND
2210 SUB NET :: CALL VCHAR(1,3,32,
     672):: SUBEND
```

Trier le fichier

Aussitôt que vous avez une dizaine de fiches, classez-les par ordre alphabétique. Ce programme va créer un index qui permettra aux autres programmes d'aller chercher les informations alphabétiquement et vous pourrez, en option, effacer des fiches. Utilisez-le fréquemment, il est en plus assez rapide.

Programme "TRIER"

Basic Etendu

```
10 !*************************
20 !TRI ALPHABETIQUE FICHIER
30 !*************************
40 DIM G$(8),D$(9),P$(450),S$(450)
50 CALL CLEAR :: CALL COLOR(13,7,7
   ):: CALL SCREEN(7)
60 CALL VCHAR(1,31,128,96):: FOR A
   =1 TO 12 :: CALL COLOR(A,5,12):
   : NEXT A
70 CALL TITRE :: CALL ENT :: CALL
   STOP
80 DISPLAY AT(23,6):"UN INSTANT SV
   P...."
90 OPEN #1:"DSK1.",INPUT ,RELATIVE
   ,INTERNAL
100 INPUT #1:T$ :: IF T$="DONNEES"
    THEN 140 ELSE IF T$<>"" THEN
    100
```

```
110 CLOSE #1 :: CALL NET :: CALL R
    IEN
120 ACCEPT AT(23,22)VALIDATE("12")
    :T
130 IF T=1 THEN 80 ELSE END
140 CLOSE #1
150 OPEN #1:"DSK1.DONNEES",RELATIV
    E,INTERNAL,FIXED 127
160 INPUT #1:TRI$,TOT$,TITRE$,IMP$
    ,G$(1),G$(2),G$(3),G$(4),G$(5)
    ,G$(6),G$(7),G$(8),NG$ :: TRI=
    VAL(TRI$):: TOT=VAL(TOT$)
170 CALL NET
180 IF TOT=0 THEN DISPLAY AT(16,2)
    :"PAS DE DONNEES SUR FICHIER":
    " ***************************"
    :: CLOSE #1 :: CALL ENT :: CAL
    L STOP :: END
190 DISPLAY AT(6,8):"OPTIONS DE TR
    I":TAB(8);"~~~~~~~~~~~~~~"
200 DISPLAY AT(10,2):"1-SUPPRIMER
    LES EFFACEMENTS": : :" 2-EFFAC
    EMENTS ET INACTIFS": : :" 3-AU
    CUNE SUPPRESSION"
210 DISPLAY AT(22,6)BEEP:"VOTRE CH
    OIX ?" :: ACCEPT AT(22,21)SIZE
    (1)VALIDATE("123"):Y :: IF Y=3
     THEN GOSUB 660 :: GOTO 400
220 !**********************
230 !*Routine d'effacement
240 !**********************
250 CALL NET :: CALL OP :: CALL DE
    P :: CALL DON(TOT)
260 DISP,C=0
270 C=C+1 :: IF C>TOT THEN GOSUB 6
    60 :: GOTO 400
280 Z=C :: GOSUB 670
290 IF D$(1)="E" OR(D$(1)="I" AND
    Y=2)THEN 310
300 IF DISP THEN Z=C-DISP :: GOSUB
     690 :: GOTO 270 ELSE 270
310 IF TOT>TRI OR C<>TOT THEN 340
320 DISP=DISP+1
330 IF C<=TRI THEN TRI=TRI-1 ELSE
    270
340 Z=TOT :: GOSUB 680 :: TOT=TOT-
    1
350 IF D$(1)="E" OR(D$(1)="I" AND
    Y=2)THEN 310
360 Z=C :: GOSUB 690 :: GOTO 270
370 !**********************
380 !* Routine de triage
390 !**********************
400 CALL NET :: CALL OP :: CALL TR
    I :: CALL DON(TOT)
410 FOR Z=1 TO TOT
420 INPUT #1,REC Z:D$(1),D$(2),D$(
    3)
430 S$(Z)=SEG$(D$(3),1,4)&SEG$(D$(
    3),POS(D$(3)," ",1)+1,1):: P$(
    Z)=STR$(Z)
440 NEXT Z :: G=1
450 G=3*G+1 :: IF G<TOT THEN 450
460 G=INT(G/3)
470 FOR I=G TO TOT :: P1$=P$(I)::
    S1$=S$(I)
480 FOR J=I-G TO 1 STEP -G :: P2$=
    P$(J):: S2$=S$(J)
490 IF S1$>=S2$ THEN 510
500 P$(J+G)=P2$ :: S$(J+G)=S2$ ::
    NEXT J
510 P$(J+G)=P1$ :: S$(J+G)=S1$
520 NEXT I
530 IF G>1 THEN 460
540 !**********************
550 !*Enregistrement index
560 !**********************
570 CLOSE #1
580 OPEN #3:"DSK1.INDEX",OUTPUT,IN
    TERNAL,FIXED 4
590 FOR I=1 TO TOT :: PRINT #3:P$(
    I):: NEXT I
600 CLOSE #3
610 DISPLAY AT(23,2):"INDEX ENREGI
    STRE.  <ENTER>"
620 CALL STOP :: END
630 !**********************
640 !* Sous programmes
650 !**********************
660 TOT$=STR$(TOT):: PRINT #1,REC
    0:TOT$,TOT$,TITRE$,IMP$,G$(1),
    G$(2),G$(3),G$(4),G$(5),G$(6),
    G$(7),G$(8),NG$ :: RETURN
670 INPUT #1,REC Z:D$(1):: RETURN
680 INPUT #1,REC Z:D$(1),D$(2),D$(
    3),D$(4),D$(5),D$(6),D$(7),D$(
    8),D$(9):: RETURN
690 PRINT #1,REC Z:D$(1),D$(2),D$(
    3),D$(4),D$(5),D$(6),D$(7),D$(
    8),D$(9):: RETURN
700 SUB TITRE :: DISPLAY AT(9,10):
    "TI 99/4A": :TAB(5);" TRI ALPH
    ABETIQUE  ": :TAB(6);" PAR IND
    EXATION " :: SUBEND
710 SUB ENT :: DISPLAY AT(23,6):"A
    PPUYEZ SUR ENTER" :: SUBEND
720 SUB RIEN :: DISPLAY AT(9,1)BEE
    P:" PAS DE FICHIER SUR DISQUE"
    :" ~~~~~~~~~~~~~~~~~~~~~~~~~":
     : :TAB(7);"OPTIONS": : :" 1.
    NOUVEL ESSAI": :" 2. SORTIE"
730 DISPLAY AT(23,7):"VOTRE CHOIX
    ?" :: SUBEND
740 SUB OP :: DISPLAY AT(8,5):"OPE
    RATION EN COURS":TAB(5);"~~~~~
    ~~~~~~~~~~~~~" :: SUBEND
750 SUB STOP
760 CALL SOUND(150,400,0)
770 CALL KEY(0,A,B):: IF NOT B THE
    N 770 :: IF A<>13 THEN 760
780 SUBEND
790 SUB NET :: CALL VCHAR(1,3,32,6
    72):: SUBEND
800 SUB DON(TOT):: CALL SOUND(150,
800,0):: DISPLAY AT(18,3):"LE NB D
    ES DONNEES VA DE:": :"
    ";1;" A ";TOT :: SUBEND
810 SUB DEP :: DISPLAY AT(16,10):"
    AU DEPART" :: SUBEND
820 SUB TRI :: DISPLAY AT(16,9):"P
    OUR LE TRI" :: SUBEND
```

Imprimer le fichier

Vous pouvez imprimer une fiche, un groupe de fiches ou la totalité du fichier. Vous pouvez imprimer un fichier organisé sous forme de colonnes. Le programme utilise une imprimante avec des caractères condensés (136/ligne), cela permet d'afficher la totalité des informations sur une ligne. Dans le cas d'une imprimante sans caractères condensés, il faudra modifier les lignes 490 à 500, 610 et 660, puis afficher sur 2 lignes chaque fiche ou bien se restreindre. Vous pouvez imprimer des étiquettes d'adresses. De nombreux paramètres vous permettront de centrer votre étiquette au mieux. J'utilise pour ma part, deux étiquettes de front (cas le plus courant) et les paramètres que j'utilise sont stockés à la ligne 110. Faites de même quand vous aurez déterminé les vôtres. Une étiquette modèle est imprimée pour déterminer le format maxi de vos données. Toutes les spécifications de l'imprimante qui sont employées dans les

Variables utilisées

Enregistrement 0 — *Nombre d'octets*

TRI$	Enregistrements triés	3 + 1
TOT$	Nb total de fiches	3 + 1
TITRE$	Option titre	1 + 1
IMP$	Nom de l'imprimante	16 + 1
D$(1 à 8)	noms des 8 groupes	88 + 8
NG$	Nb de groupes utilisés	3 + 1

Les variables numériques TRI, TOT, NG, sont stockées en tant que chaîne pour qu'elles n'occupent que 3 octets maximum (nb de 3 chiffres), au lieu de 8 pour une variable numérique.

Autres enregistrements

D$(1)	A (actif) ou I (inactif) ou E (effacement)
D$(2)	8 "N" ou "O" selon que la personne se situe ou non dans un des 8 groupes. Par exemple "ONONNNNN" pour une personne faisant partie des groupes 1 et 8.
D$(3)	Nom, Prénom
D$(4)	Adresse
D$(5)	Code postal
D$(6)	Ville
D$(7)	Titre
D$(8)	Divers

différents programmes sont expliquées dans le programme *TITRER EN GROS*.

Programme "IMPRIMER"

Basic étendu

```
5 !*************************
6 !* IMPRIMER LE FICHIER
7 !*************************
10 ON WARNING NEXT :: OPTION BASE
   1 :: ON BREAK NEXT
20 DIM G$(8),D$(9),ETIQ$(4,3),P$(4
   50)
30 INACTIF$="N" :: BLANC$=RPT$(" "
   ,40)
40 CALL CLEAR :: CALL SCREEN(7)
50 CALL COLOR(13,7,7)
60 CALL VCHAR(1,31,129,96)
70 FOR I=1 TO 12
80 CALL COLOR(I,5,12)
90 NEXT I
100 CALL TITRE :: CALL ENT
110 NCOL=80 :: LARG=36 :: ESP=5 ::
    NBET=2 :: DEP,NREPS,RECNO,ENR
   =1
120 CALL STOP
130 DISPLAY AT(23,6):"UN INSTANT S
   VP...."
140 OPEN #1:"DSK1.",INTERNAL,RELAT
   IVE,INPUT
150 INPUT #1:T$ :: IF T$="DONNEES"
    THEN 190 ELSE IF T$<>"" THEN
    150
160 CLOSE #1 :: CALL NET :: CALL R
   IEN
170 DISPLAY AT(23,6):"VOTRE CHOIX
   ?" :: ACCEPT AT(23,20)VALIDATE
   ("12")SIZE(1):T
180 IF T=2 THEN 900 ELSE 130
190 CLOSE #1
200 GOSUB 920 :: GOSUB 1850
210 CALL NET :: CALL DISP(IMP$)
220 ACCEPT AT(14,22)VALIDATE("ON")
    SIZE(-1):A$
230 IF A$="N" THEN 250
240 RESTORE 2020 :: READ A$ :: INP
   UT A$:IMP$ :: GOSUB 940
250 CALL NET :: CALL OPT
260 DISPLAY AT(23,6)BEEP:"VOTRE CH
   OIX ?" :: ACCEPT AT(23,20)SIZE
   (1)VALIDATE("1234"):A
270 ON A GOTO 290,450,710,900
275 !*************************
280 !* Toute information
285 !*************************
290 OPEN #9:IMP$,OUTPUT :: GOSUB 1
   450
300 CALL NET :: CALL IMP
310 PRINT #9:"  FICHIER D'ADRESSES
   ":"  =================": :
320 FOR RECNO=1 TO TOT :: GOSUB 97
   0
330 IF M<=12 THEN 410
340 IF TITRE$="O" THEN PRINT #9:D$
   (8)
350 PRINT #9:D$(3):D$(4):D$(5);" "
   ;D$(6):D$(7):D$(9)
360 PRINT #9
370 FOR I=1 TO 8 :: IF SEG$(D$(2),
   I,1)="N" THEN 390
380 PRINT #9:TAB(5);G$(I)
390 NEXT I
400 PRINT #9
410 NEXT RECNO
420 GOSUB 1620
430 CLOSE #9 :: GOTO 250
435 !*************************
440 !* Fichier organise
445 !*************************
450 OPEN #9:IMP$&".CR",OUTPUT :: G
   OSUB 1450
460 INPUT "TITRE ET DATE: ":DA$ ::
    CALL NET :: CALL IMP
470 PRINT #9:DA$;CHR$(10);CHR$(10)
480 RESTORE 2030 :: READ A$,A1$,A2
   $,A3$,A4$
490 PRINT #9:CHR$(27);"C";CHR$(16)
   ;"004";A$;
500 PRINT #9:CHR$(16);"033";A1$;CH
   R$(16);"066";A2$;CHR$(16);"090
   ";A3$;CHR$(16);"105";A4$;CHR$(
   10)
510 READ A$,A1$,A2$,A3$,A4$
520 PRINT #9:CHR$(16);"004";A$;
530 PRINT #9:CHR$(16);"033";A1$;CH
   R$(16);"066";A2$;CHR$(16);"090
   ";A3$;CHR$(16);"105";A4$;CHR$(
   10)
540 DISPLAY AT(20,2):"VERIFIEZ L'A
   LIGNEMENT": :"
   <ENTER>"
550 ACCEPT AT(22,24):A$ :: DISPLAY
    AT(20,2):: DISPLAY AT(22,10):
   : A=0
560 FOR I=1 TO TRI :: RECNO=VAL(P$
   (I))
570 PRINT #9:CHR$(24):: GOSUB 970
580 IF M<12 THEN 620
590 A=A+1
600 PRINT #9,USING "### ":A;:: PRI
   NT #9:D$(3);
610 PRINT #9:CHR$(16);"033";D$(4);
   CHR$(16);"066";D$(5);"    ";D$(
   6);CHR$(16);"090";D$(7);CHR$(1
   6);"105";D$(9);CHR$(10)
620 NEXT I
630 IF TOT=TRI THEN 680
640 FOR RECNO=TRI+1 TO TOT :: PRIN
   T #9:CHR$(24):: GOSUB 970 :: I
   F M<12 THEN 670 :: A=A+1
650 PRINT #9,USING "### ":A;:: PRI
   NT #9:D$(3);
660 PRINT #9:CHR$(16);"033";D$(4);
   CHR$(16);"066";D$(5);"    ";D$(
   6);CHR$(16);"090";D$(7);CHR$(1
   6);"105";D$(9);CHR$(10)
670 NEXT RECNO
680 GOSUB 1620
690 A=0 :: PRINT #9:CHR$(27);"N" :
   : CLOSE #9 :: GOTO 250
695 !*************************
700 !Impression d'etiquette
705 !*************************
710 OPEN #9:IMP$,OUTPUT :: GOSUB 1
   450
720 RESTORE 1910 :: CALL NET
730 DISPLAY AT(3,1):"~~~~~~~~~~~~~
   ~~~~~~~~~~~~~~":"  OPTIONS DE
    MISE EN PLACE":"      DES ETI
   QUETTES":"~~~~~~~~~~~~~~~~~~~~
   ~~~~~~~~"
740 DISPLAY AT(11,1)BEEP:" CARACTE
   RES GRAS (O/N)? O" :: ACCEPT A
   T(11,25)VALIDATE("ON")SIZE(-1)
   :A$ :: IF A$="O" THEN EP$="#"
   ELSE EP$="N"
750 MOINS=32 :: PLUS=132 :: VAR=NC
   OL :: GOSUB 1770 :: NCOL=VAR
760 MOINS=1 :: PLUS=4 :: VAR=NBET
   :: GOSUB 1770 :: NBET=VAR
770 MOINS=0 :: PLUS=5 :: VAR=DEP :
   : GOSUB 1770 :: DEP=VAR
780 MOINS=28 :: PLUS=40 :: VAR=LAR
   G :: GOSUB 1770 :: LARG=VAR
790 IF NBET*LARG<=NCOL-NBET*DEP TH
   EN 810
800 CALL NET :: CALL TROP :: CALL
   ENT :: CALL STOP :: GOTO 720
810 MOINS=1 :: PLUS=20 :: VAR=NREP
   S :: GOSUB 1770 :: NREPS=VAR
820 MOINS=0 :: PLUS=10 :: VAR=ESP
   :: GOSUB 1770 :: ESP=VAR
830 GOSUB 1150 :: CALL NET
840 CALL IMP :: FOR RECNO=1 TO TOT
    :: GOSUB 970 :: GOSUB 1680 ::
    NEXT RECNO
850 IF ENR=1 THEN 890
860 FOR J=ENR TO NBET
870 FOR K=1 TO 3 :: ETIQ$(J,K)=""
   :: NEXT K
880 NEXT J :: GOSUB 1320
890 PRINT #9:CHR$(27);"$" :: CLOSE
    #9 :: GOTO 250
900 CLOSE #1 :: END
905 !--------------------
910 ! s/p lect/enr fichier
915 !--------------------
920 OPEN #1:"DSK1.DONNEES",RELATIV
   E,INTERNAL,FIXED 127
930 INPUT #1,REC 0:TRI$,TOT$,TITRE
   $,IMP$,G$(1),G$(2),G$(3),G$(4)
   ,G$(5),G$(6),G$(7),G$(8),NG$
935 TRI=VAL(TRI$):: TOT=VAL(TOT$):
   : NG=VAL(NG$):: RETURN
940 TRI$=STR$(TRI):: TOT$=STR$(TOT
   ):: NG$=STR$(NG)
945 PRINT #1,REC 0:TRI,TOT,TITRE$,
   IMP$,G$(1),G$(2),G$(3),G$(4),G
   $(5),G$(6),G$(7),G$(8),NG :: R
   ETURN
950 !s/p saisie+tri enreg.
955 !--------------------
960 RECNO=VAL(PT$(Y))
970 INPUT #1,REC RECNO:D$(1),D$(2)
   ,D$(3),D$(4),D$(5),D$(6),D$(7)
   ,D$(8),D$(9)
980 CALL KEY(0,K,S):: IF NOT S THE
   N 1070
990 IF K=15 THEN RECNO=TOT+1 :: RE
   TURN
1000 IF K<>32 THEN 1070
1010 CALL SCREEN(11)
1020 CALL KEY(0,K,S)
1030 IF (S=1)*(K=32)THEN 1060
1040 CALL SOUND(-1,1000,12)
1050 GOTO 1020
1060 CALL SCREEN(7)
1070 M=1
1080 IF D$(1)="E" THEN RETURN
1090 IF INACTIF$="N" THEN 1110
1100 IF D$(2)="NNNNNNNN" THEN M=13
    :: RETURN
1110 FOR M=1 TO NG
1120 IF SEG$(D$(2),M,1)=SEG$(GROUP
   $,M,1)THEN M=12
1130 NEXT M :: RETURN
1135 !--------------------
1140 !Format de l'etiquette
1145 !--------------------
1150 CALL NET :: CALL METTRE :: CA
   LL ENT :: CALL STOP
1160 RESTORE 1990 :: FOR I=1 TO 3
    :: READ ETIQ$(1,I)
1170 ETIQ$(2,I)=ETIQ$(1,I)
1180 ETIQ$(3,I)=ETIQ$(1,I)
1190 ETIQ$(4,I)=ETIQ$(1,I)
1200 NEXT I
1210 ON NBET GOTO 1250,1240,1230,1
   220
1220 GOSUB 1310
1230 GOSUB 1310
1240 GOSUB 1310
1250 GOSUB 1310
1260 CALL ALIGN
1270 ACCEPT AT(23,26)VALIDATE("ON"
   )SIZE(-1):A$
1280 IF A$="N" THEN 1210
1290 RETURN
1295 !--------------------
1300 !Routine d'impression
1305 !--------------------
1310 IF NBET>ENR THEN ENR=ENR+1 ::
    RETURN
1320 PRINT #9:CHR$(27);EP$ :: DEP$
   =RPT$(" ",DEP)
1330 FOR I=1 TO 3
1340 FOR J=1 TO NBET
1350 PRINT #9:DEP$&SEG$(ETIQ$(J,I)
   &BLANC$,1,LARG);
1360 NEXT J
1370 NEXT I
1380 PRINT #9
1390 ENR=1
1400 FOR I=1 TO ESP
1410 PRINT #9
1420 NEXT I
1430 RETURN
1435 !--------------------
1440 !Choix groupes a impr.
1445 !--------------------
1450 RESTORE 1970 :: CALL NET
1460 IF NG=0 THEN GROUP$="00000000
   " :: GOTO 1570
1470 READ A$ :: DISPLAY AT(7,1):A$
   :"  ~~~~~~~~~~~~~~~~~"
1480 READ A$ :: GROUP$=""
1490 FOR I=1 TO 8
1500 IF (G$(I)="")+(G$(I)=" ")THEN
```

```
     1540
1510 DISPLAY AT(9+I,3)BEEP:G$(I);T
     AB(15);A$
1520 ACCEPT AT(9+I,20)VALIDATE("ON
     ")SIZE(-1):GR$
1530 IF GR$="O" THEN 1550
1540 GR$="0"
1550 GROUP$=GROUP$&GR$
1560 NEXT I
1570 DISPLAY AT(10+I,3):"INACTIFS"
     ;TAB(15);"(O/N)N"
1580 ACCEPT AT(10+I,20)VALIDATE("O
     N")SIZE(-1):INACTIF$
1590 IF INACTIF$="O" THEN RETURN
1600 IF GROUP$="00000000" THEN 250
     ELSE RETURN
1605 !-------------------
1610 ! Fin d'impression
1615 !-------------------
1620 RESTORE 2050 :: READ A$,B$
1630 PRINT #9:CHR$(10);CHR$(10)
1640 PRINT #9:"  ";A$;CHR$(10);"
     ";B$
1650 PRINT #9:CHR$(10);CHR$(10)
1660 RETURN
1665 !-------------------
1670 ! Impression etiquette
1675 !-------------------
1680 IF M<12 THEN RETURN
1690 FOR K=1 TO NREPS
1700 ETIQ$(ENR,1)=D$(8)&" "&D$(3)
1710 ETIQ$(ENR,2)=D$(4)
1720 ETIQ$(ENR,3)=D$(5)&" "&D$(6)
1730 GOSUB 1310
1740 NEXT K
1750 RETURN
1755 !-------------------
1760 !Saisie options etiq.
1765 !-------------------
1770 READ A$,T$,B$ :: DISPLAY AT(1
     1,2):A$: : :TAB(2);VAR: : :T$
1780 ACCEPT AT(17,23)VALIDATE("ON"
     )SIZE(-1)BEEP:A$
1790 IF A$="N" THEN 1830
1800 DISPLAY AT(23,1):B$
1810 ACCEPT AT(23,25)SIZE(-3)VALID
     ATE(DIGIT)BEEP:VAR
1820 IF VAR<MOINS OR VAR>PLUS THEN
     1810
1830 CALL NET2 :: RETURN
1835 !-------------------
1840 ! Fichier index
1845 !-------------------
1850 OPEN #3:"DSK1.INDEX",INTERNAL
     ,INPUT ,FIXED 4
1860 FOR I=1 TO TRI
1870 INPUT #3:P$(I)
1880 NEXT I
1890 CLOSE #3
1900 RETURN
1905 !-------------------
1906 ! Zone des datas
1907 !-------------------
1910 DATA "NB DE COLONNES MAXIMUM"
     ," MODIFICATION  (O/N)? N","
     NOUVEAU NOMBRE(32-132)?"
1920 DATA "NB D'ETIQUETTES PAR LIG
     NE"," MODIFICATIONS  (O/N)? N"
     ," NOUVEAU NOMBRE  (1-4)  ? "
1930 DATA "MARGE DE GAUCHE"," MODI
     FICATIONS  (O/N)? N"," NOUVEAU
      NOMBRE  (0-5)  ?"
1940 DATA "LARGEUR DE L'ETIQUETTE"
     ," MODIFICATIONS  (O/N)? N","N
     OUVELLE LARGEUR(28-40)?"
1950 DATA "NB DE CHAQUE EXEMPLAIRE
     "," MODIFICATIONS (O/N)? N","
      NOUVEAU NOMBRE  (1-20)?"
1960 DATA "ECCART ENTRE ETIQUETTES
     "," MODIFICATIONS (O/N)? N","
      NOUVEAU NOMBRE  (0-10)?"
1970 DATA "  GROUPES A IMPRIMER"
1980 DATA (O/N)O
1990 DATA "TITRE..NOM...PRENOM....
     ......"
2000 DATA "ADRESSE................
     ......"
2010 DATA "CODE.POSTAL....VILLE...
     ......"
2020 DATA " NOUVEAU DISPOSITIF ? "
2030 DATA "NOM        PRENOM",ADRESS
     E,CODE       VILLE,"  TEL","
       DIVERS"
2040 DATA ----------------------,
     --------------------------,-
     ----  ----------,-----------
     ,----------------------
2050 DATA ----------------,FIN D'I
     MPRESSION
2055 !-------------------
2056 ! s/programmes fermes
2057 !-------------------
2060 SUB TITRE :: DISPLAY AT(9,10)
     :"TI 99/4A": :TAB(4);"IMPRESS
     ION DU FICHIER":TAB(4);"~~~~~
     ~~~~~~~~~~~~~~~~" :: SUBEND
2070 SUB ENT :: DISPLAY AT(23,6):"
     APPUYEZ SUR ENTER" :: SUBEND
2080 SUB RIEN :: DISPLAY AT(8,2):"
     PAS DE FICHIER SUR DISQUE": :
     :TAB(9);"OPTIONS": : :"
     1. NOUVEL ESSAI": :"      2. S
     ORTIE" :: SUBEND
2090 SUB OPT :: DISPLAY AT(6,4):"O
     PTIONS D'IMPRESSION":TAB(4);"
     ~~~~~~~~~~~~~~~~~~~~
2100 DISPLAY AT(10,3):"1. TOUT LE
     FICHIER": :TAB(3);"2. FICHIER
      ORGANISE": :TAB(3);"3. ETIQU
     ETTES": :TAB(3);"4. SORTIE" :
     : SUBEND
2110 SUB TROP :: DISPLAY AT(11,6):
     "LIGNE D'IMPRESSION": :TAB(9)
     ;"TROP LONGUE" :: SUBEND
2120 SUB METTRE :: DISPLAY AT(8,5)
     :"METTEZ LES ETIQUETTES": :"
          SUR L'IMPRIMANTE" :: SUB
     END
2130 SUB ALIGN :: DISPLAY AT(23,2)
     :"ALIGNEMENT CORRECT(O/N)?O"
     :: SUBEND
2140 SUB DISP(IMP$):: DISPLAY AT(8
     ,1):" DISPOSITIF D'IMPRESSION
     ": : :TAB(3);IMP$
2150 DISPLAY AT(14,1)BEEP:" MODIFI
     CATIONS  (O/N)?N" :: SUBEND
2160 SUB IMP :: DISPLAY AT(12,1):"
      L'IMPRESSION EST EN COURS":"
     ~~~~~~~~~~~~~~~~~~~~~~~~" :
     : SUBEND
2170 SUB NET :: CALL VCHAR(1,3,32,
     672):: SUBEND
2180 SUB STOP
2190 CALL SOUND(150,800,8)
2200 CALL KEY(0,K,S):: IF NOT S TH
     EN 2200 ELSE IF K<>13 THEN 21
     90
2210 SUBEND
2220 SUB NET2 :: FOR I=11 TO 23 ST
     EP 3 :: DISPLAY AT(I,1):: NEX
     T I :: SUBEND
```

Recherche sur un fichier

Vous pouvez afficher la totalité des noms en appuyant seulement sur <enter> à la question <début du nom à rechercher> sinon, le programme affichera tous les noms commençant par la (ou les) lettre(s) choisie(s) (6 maximum). Ce programme, comme les autres, tourne plus rapidement si le fichier est classé alphabétiquement par index.

Programme "RECHERCHER"

Basic étendu

```
5 !**********************
6 ! PROGRAMME DE RECHERCHE
7 !**********************
10 ON BREAK NEXT
20 CALL CLEAR
30 DIM P$(450),D$(4)
40 CALL SCREEN(7):: CALL COLOR(13,
   11,1)
50 CALL VCHAR(1,31,129,96)
60 FOR A=1 TO 12 :: CALL COLOR(A,5
   ,12):: NEXT A
70 CALL TITRE :: CALL ENT :: CALL
   STOP(K)
80 DISPLAY AT(23,6):"UN INSTANT SV
   P..."
90 OPEN #1:"DSK1.",RELATIVE,INTERN
   AL,INPUT
100 INPUT #1:T$
110 IF T$="DONNEES" THEN 160 ELSE
    IF T$<>"" THEN 100
120 CLOSE #1
130 CALL NET :: CALL RIEN :: CALL
    CHOIX
140 ACCEPT AT(22,22)SIZE(-1)VALIDA
    TE("12")BEEP:I
150 IF I=2 THEN 390 ELSE 80
160 CLOSE #1
170 OPEN #1:"DSK1.DONNEES",RELATIV
    E,INTERNAL,FIXED 127
180 INPUT #1:TRI$,TOT$ :: TRI=VAL(
    TRI$):: TOT=VAL(TOT$)
190 OPEN #3:"DSK1.INDEX",INPUT ,IN
    TERNAL,FIXED 4 :: FOR I=1 TO T
    RI
200 INPUT #3:P$(I)
210 NEXT I :: CLOSE #3
220 IF TOT=0 THEN 380
230 CALL NET :: CALL RECH
240 ACCEPT AT(24,2)VALIDATE(UALPHA
    ):T$ :: IF T$="" THEN 290
250 L=LEN(T$)
260 IF L<6 THEN 450
270 CALL NET :: CALL TROP :: CALL
    ENT :: CALL STOP(K):: GOTO 230
290 CALL NET
300 FOR I=1 TO TOT :: IF I<=TRI TH
    EN Z=VAL(P$(I))ELSE Z=I
310 GOSUB 400
320 PRINT USING "###":I;:: PRINT "
    "&D$(3)
330 NEXT I
340 CALL HCHAR(12,1,42,2)
350 CALL HCHAR(12,31,42,2)
360 CALL FIN :: CALL STOP(K)
370 CALL VCHAR(1,31,129,96):: IF K
    =15 THEN 230
380 CLOSE #1
390 END
395 !********************
400 INPUT #1,REC Z:D$(1),D$(2),D$(
    3):: CALL KEY(0,K,S):: IF K=15
     THEN 230 ELSE IF NOT S THEN R
    ETURN
410 CALL SCREEN(11)
420 CALL KEY(0,K,S):: IF S<>1 THEN
     CALL SOUND(-1,1000,10):: GOTO
     420
430 CALL SCREEN(7):: IF K<>32 THEN
     230
440 RETURN
450 DISPLAY AT(21,1):" NOM RECHERC
    HE:":" ~~~~~~~~~~~~" :: I=1 :
    : J=TRI
460 ZT=INT((I+J)/2):: Z=VAL(P$(ZT)
    )
470 INPUT #1,REC Z:D$(1),D$(2),D$(
    3):: D$(4)=SEG$(D$(3),1,L)
480 IF D$(4)=T$ THEN 530
490 IF D$(4)<T$ THEN I=ZT+1 ELSE J
    =ZT-1
500 IF (ZT<1)+(I>J)=0 THEN 460 ELS
    E 570
510 INPUT #1,REC Z:D$(1),D$(2),D$(
    3):: D$(4)=SEG$(D$(3),1,L)
520 IF D$(4)<>T$ THEN 540
530 IF ZT>1 THEN ZT=ZT-1 :: Z=VAL(
    P$(ZT)):: GOTO 510
540 INPUT #1,REC Z:D$(1),D$(2),D$(
    3):: D$(4)=SEG$(D$(3),1,L)
550 IF D$(4)=T$ THEN 560 ELSE ZT=Z
    T+1 :: Z=VAL(P$(ZT)):: GOTO 54
```

```
    0
560 IF D$(4)=T$ THEN PRINT USING "
    ### ":ZT;:: PRINT D$(3)ELSE 57
    0
565 IF ZT<>TRI THEN ZT=ZT+1 :: Z=V
    AL(P$(ZT)):: INPUT #1,REC Z:D$
    (1),D$(2),D$(3):: D$(4)=SEG$(D
    $(3),1,L):: GOTO 560
570 IF TRI=TOT THEN 610
580 FOR Z=TRI+1 TO TOT :: INPUT #1
    ,REC Z:D$(1);D$(2);D$(3)
590 D$(4)=SEG$(D$(3),1,L):: IF D$(
    4)=T$ THEN PRINT USING "### ":
    Z;:: PRINT D$(3)
600 NEXT Z
610 CALL FIN :: CALL STOP(K):: IF
    K=15 THEN 230 ELSE STOP
620 SUB TITRE :: DISPLAY AT(10,10)
    :"TI 99/4A": :TAB(4);"RECHERCH
    E SUR FICHIER":TAB(4);"~~~~~~~
    ~~~~~~~~~~~~~~" :: SUBEND
630 SUB ENT :: DISPLAY AT(23,6):"A
    PPUYEZ SUR ENTER" :: SUBEND
635 SUB RIEN :: DISPLAY AT(8,6):"A
    BSENCE DE FICHIER":TAB(6);"~~~
    ~~~~~~~~~~~~~~~": : :TAB(8);"O
    PTIONS"
640 DISPLAY AT(14,5):"1. NOUVEL ES
    SAI": :TAB(5);"2. SORTIE" :: S
    UBEND
650 SUB CHOIX :: DISPLAY AT(22,8):
    "VOTRE CHOIX ?" :: SUBEND
660 SUB RECH :: DISPLAY AT(22,1)BE
    EP:" DEBUT DU NOM A RECHERCHER
    " :: SUBEND
670 SUB TROP :: DISPLAY AT(12,1)BE
    EP:" IL Y A TROP DE LETTRES !!
    ": :"        (5 MAXIMUM)" :: SUB
    END
680 SUB NET :: CALL VCHAR(1,3,32,6
    72):: SUBEND
690 SUB STOP(K)
700 CALL SOUND(150,800,2)
710 CALL KEY(0,K,S):: IF NOT S THE
    N 710
720 IF K<>15 AND K<>13 THEN 700
730 SUBEND
740 SUB FIN :: PRINT :" <BACK>ou<E
    NTER>pour arret" :: SUBEND
```

Titrer en gros

Ce programme dépend directement des possibilités de votre imprimante. A cet effet, il comporte une bibliothèque de toutes les spécifications employées dans les différents programmes afin que vous puissiez faire une adaptation. Le titre est en double largeur, caractères italiques, impression en gras. Une fois terminé, le sommaire est rechargé automatiquement.

Programme "TITRER"

Basic étendu

```
5 !************************
6 !*  TITRER EN GROS
7 !************************
10 CALL CLEAR :: CALL SCREEN(7)::
   CALL COLOR(13,7,7)
20 CALL VCHAR(1,31,129,96)
30 FOR I=1 TO 12 :: CALL COLOR(I,5
   ,12):: NEXT I
40 DISPLAY AT(8,10):"TI 99/4A": :
   :"  TITRER EN GROS CARACTERES":
   "  ~~~~~~~~~~~~~~~~~~~~~~~~~"
45 DISPLAY AT(17,3):"<FCTN AID>:sp
   ecifications": :"      <ENTER>:c
   ontinuer"
50 CALL SOUND(150,800,5)
60 CALL KEY(0,K,S):: IF NOT S THEN
    60 ELSE IF K=13 THEN 120 ELSE
   IF K<>1 THEN 60
65 CALL NET :: DISPLAY AT(1,2):"IM
   PRIMANTE SEIKOSHA 550 A":" ~~~~
   ~~~~~~~~~~~~~~~~~~~~~~"
70 DISPLAY AT(4,1):"CHR$(14) carac
   tere dilate":"CHR$(15) fin de d
   ilatation":"CHR$(24) vide le bu
   ffer"
80 DISPLAY AT(7,1):"CHR$(16);""XXX
   "" tabulation":"CHR$(10) retour
    chariot": : :"CHR$(27);annonce
    les donnees":"""C"" condenses
   (136/ligne)"
90 DISPLAY AT(13,1):"""B"" caracte
   res italique":"""#"" caracteres
    gras":"""$"" fin de carateres
   gras":"""N"" caracteres normaux
   ": :"IMP$&"".CR""?"
100 DISPLAY AT(20,1):""".CR"" supp
    rime le retour du chariot au 8
    0 Ieme caractere(pour en mettr
    e jusqu'a 136)": :"   <enter> p
    our continuer"
110 CALL KEY(0,K,S):: IF NOT S THE
    N 110
120 CALL NET :: OPEN #9:"PIO"
130 DISPLAY AT(8,2)BEEP:"TITRE EN
    GROS CARACTERES":" ~~~~~~~~~~
    ~~~~~~~~~~~~~": : :" Donnee: 1
     ligne maximum"
140 ACCEPT AT(16,1)SIZE(-28):A$ ::
     IF A$="" THEN 210
150 DISPLAY AT(20,4):"IMPRESSION E
    N COURS":TAB(4);"~~~~~~~~~~~~
    ~~~~~~"
160 PRINT #9:CHR$(27);"B";CHR$(14)
170 X=INT((40-LEN(A$))/2)
180 PRINT #9:CHR$(27);"#";TAB(X);A
    $
190 PRINT #9:TAB(X-2);RPT$("~",LEN
    (A$))
200 PRINT #9:CHR$(15);CHR$(27);"$"
    ;CHR$(27);"N";: :
210 CLOSE #9
220 RUN "DSK1.LOAD"
230 SUB NET :: CALL VCHAR(1,3,32,6
    72):: SUBEND
```

Programmes en assembleur

Saisie

Lorsque vous entrerez ces programmes, NE SAISISSEZ PAS LES COMMENTAIRES, car vous ne pourriez pas les assembler sur une seule face de disquette. Si vous disposez de plusieurs lecteurs, aucun problème ; il suffira d'envoyer le fichier objet sur une autre disquette. Le programme est prêt à être transformé en "image mémoire" grâce au programme "SAVE" de la disquette Editeur/Assembleur Texas Instruments (une fois assemblé), il se chargera alors environ trois fois plus vite. Si vous l'utilisez avec la Mini-mémoire, qui ne permet pas de charger ce type de fichier, rajoutez "FICHES" après le END du fichier source "FICH0" pour que votre programme démarre automatiquement.

Initialiser le fichier

C'est le premier programme à faire tourner. Il permet d'ouvrir les fichiers et de contrôler si l'espace initial pour les données est suffisant. Vous devrez alors rentrer le nom des groupes dans lequels vous mettrez vos fiches (au moins une) et indiquer le nom de votre imprimante (ce n'est pas obligatoire). Vous pourrez modifier ce nom et les paramètres éventuels à chaque fois que vous choisirez l'option "IMPRESSION".

Gestion du fichier

C'est le programme qui permet de saisir les données et, éventuellement, de modifier les noms de groupes ou les informations contenues dans les fiches. Il permet également de marquer une fiche pour l'effacer (avec "TRI"), de rechercher une fiche ou un groupe et de l'afficher sur l'écran. Chaque enregistrement contient les informations relatives à l'adresse, plus un caractère : soit A (actif), I (inactif) ou E (effacement), précédant les huit octets (groupe dans lequel est la personne). Evitez de modifier les saisies concernant les noms, prénoms, adresses, codes, villes et numéros de téléphone. En revanche, les autre informations sont spécifiques et vous pourrez les changer à votre convenance.

Source "FICH0"

```
       COPY "DSK1.FICH1"
       COPY "DSK1.FICH2"
       COPY "DSK1.FICH3"
       COPY "DSK1.FICH4"
SLAST  END   ****** rajouter ici FICHES
* si vous ne voulez utiliser que le
* fichier classique avec depart automatique
```

Pour effectuer des modifications, il faut, dans "PROGRAMME DE GESTION" et à l'étiquette "TEXT6", changer l'intitulé des saisies. Dans "INITIALISER LE FICHIER", à l'étiquette "NBSAIS", il faut éventuellement les nouveaux chiffres, chaque chiffre indiquant le nombre maximum de chaque

rubrique (nom, adresse, code, ville, téléphone, etc...) dans l'ordre ou vous les avez mis dans "TEXT6", le total ne devant pas dépasser 127. Dans "IMPRIMER LE FICHIER", à l'étiquette "ENTET1" et "ENTET2", modifier les intitulés. Dans le même fichier, mais cette fois à l'étiquette "IMPR5" vous trouverez les paramètres d'impression :

- TLIGNE correspond à l'imprimante ;
- TPAB correspond à l'emplacement où se trouve l'information sur l'enregistrement.

Trier le fichier

Dès que vous avez une dizaine de fiches, classez-les par ordre alphabétique. Ce programme va créer un index qui permettra aux autres programmes d'aller chercher les informations alphabétiquement et vous pourrez, en option, effacer des fiches. Utilisez-le fréquemment ; il est très rapide. Vous pourrez également trier les autres informations, sauf les adresses et les chiffres d'inégales longueur, car le programme ne prend en compte que quatre octets au maximum.

Imprimer le fichier

Vous pouvez organiser un fichier organisé sous forme de colonnes. Le choix des groupes à imprimer est aussi possible. Le programme utilise une imprimante avec caractères condensés (136 caractères par lignes) ; cela permet l'affichage de la totalité des informations sur une ligne. Dans le cas d'une imprimante sans ce type de caractère, il faudra modifier, dans "IMPR5" comme indiqué plus haut, les saisies dans TPAB, leur report dans "TLIGNE", et se restreindre à un nombre de caractères plus limité.
Vous pouvez imprimer des étiquettes d'adresses ; le programme est conçu pour une imprimante 80 colonnes pouvant imprimer deux étiquettes de front.

De nombreux paramètres vous permettront de centrer vos étiquettes aux mieux. J'utilise pour ma part deux étiquettes de front (mais vous pouvez n'en utiliser qu'une) et les paramètres que j'utilise sont stockés dans le fichier "ETIQUETTES D'ADRESSES" aux étiquettes MG (marge de gauche), ENTRE (écart entre les deux étiquettes), SAUT (nombre de lignes à sauter entre les séries), et enfin MAXI (nombre maximum de lignes de l'imprimante). Il y a également, à l'étiquette HEX10, des codes LF "CHR$(10)" commun à toutes les imprimantes, et qui provoquent l'impression de la mémoire tampon, un retour chariot et un saut de ligne.

Une étiquette modèle est imprimée pour déterminer le format maximum de vos données. Toutes les spécifications de l'imprimante qui sont employées sont dans "IMPRESSION DU FICHIER", aux étiquettes COMPR (caractères compressés), BIGCAR (gros caractères + caractères gras) et FINBIG (fin gros caractères, fin caractères gras et remise en mode normal).

A vous de placer les codes correspondants pour votre imprimante. Dans le cas où vous n'auriez pas toutes ces possibilités, il faudra mettre des >00 à la place des codes manquants ; de cette façon, vous n'aurez rien d'autre à modifier.

Source "FICH1"

```
       DEF  FICHES,SFIRST,SLOAD,SLAST
       REF  VSBW,VMBW,VMBR,KSCAN
       REF  DSRLNK,VWTR,XMLLNK,VSBR
       AORG >A000
SFIRST
SLOAD  B    @FICHES
TEXTE  TEXT -'XXXXXXXXXXXXXXXXXXXXXXX '
       TEXT -'X  FICHIER D''ADRESSES  X '
       TEXT -'XXXXXXXXXXXXXXXXXXXXXXX '
       TEXT -'SOMMAIRE '
       TEXT -'~~~~~~~~ '
       TEXT -'**************************** '
       TEXT -'*                          * '
       TEXT -'* Appuyez                  * '
       TEXT -'*   sur                    * '
       TEXT -'*    1  pour  Gerer        * '
       TEXT -'*    2        Imprimer     * '
       TEXT -'*    3        Initialiser  * '
       TEXT -'*    4        Trier        * '
       TEXT -'*    5        Terminer     * '
       TEXT -'*                          * '
       TEXT -'**************************** '
ADDRE  DATA 48,88,128,255,295,406,446
       DATA 486,526,566,606,646,686,726,766
       DATA 806,0
REG    EQU  >20BA
TPAB   EQU  >1000
PAB    EQU  >F80
PAB2   EQU  >F00
RTCLA  EQU  >8375
PNTR   EQU  >8356
* mode input int/fix 38
DSK0   DATA >000D,>1000,>2626,>0000,>0005
       TEXT 'DSK1.'
* mode input int/fix 127
DSK1   DATA >000D,>1000,>7F7F,>0000,>000C
       TEXT 'DSK1.DONNEES'
* mode input int/var 254
DSK2   DATA >000C,>1000,>FEFE,>0000,>000A
       TEXT 'DSK1.INDEX'
* emplacement des noms de groupes
GROUPE DATA >2020,>2020,>2020,>2020,>2020
       DATA >2020,>2020,>2020
       DATA >2020,>2020,>2020,>2020,>2020
       DATA >2020,>2020,>2020
       DATA >2020,>2020,>2020,>2020,>2020
       DATA >2020,>2020,>2020
       DATA >2020,>2020,>2020,>2020,>2020
       DATA >2020,>2020,>2020
       DATA >2020,>2020,>2020,>2020,>2020
       DATA >2020,>2020,>2020
       DATA >2020,>2020,>2020,>2020
TRI    DATA 0
TOT    DATA 0
NGR    DATA 0
A      BYTE 65
E      BYTE 69
I      BYTE 73
ZERO   BYTE 48
UN     BYTE 49
DEUX   BYTE 50
TROIS  BYTE 51
```

```
QUATRE BYTE 52
CINQ   BYTE 53
NEUF   BYTE 57
ENT    BYTE 13
OUI    BYTE 79
NON    BYTE 78
BACK   BYTE 15
REDO   BYTE 6
FCTNS  BYTE 8
FCTND  BYTE 9
INDX   DATA 0
TERM   DATA 0
DRAP   DATA 0
DRAP1  DATA 0
DRAP2  DATA 0
D450   DATA 450
DIX    DATA 10
SECT   DATA 0
LGR    DATA 0
NLIG   DATA 0
LSAIS  DATA 0
D40    DATA 40
MINI   DATA 227
D320   DATA 320
D11    DATA 11
D33    DATA 33
D80    DATA 80
ATTRIB DATA >0100
TAMPON BSS  118
BUFGR  BYTE 0
       BSS  8
       EVEN
```

* s/p initialisation "tampon"

```
INITAM CLR  R2
TA     MOVB R1,@TAMPON(R2)
       INC  R2
       CI   R2,118
       JNE  TA
       RT
```

* s/p efface ecran

```
CLEAR1 LI   R0,160
       JMP  CL1
CLEAR  CLR  R0
CL1    LI   R1,>2000
CL     BLWP @VSBW
       INC  R0
       CI   R0,960
       JNE  CL
       RT
```

* s/p affichage

```
AFFICH MOV  *R3+,R0
       JEQ  RET
AFF    MOVB *R2+,R1
       JLT  AFFICH
       BLWP @VSBW
       INC  R0
       JMP  AFF
RET    RT
```

* s/p fermeture fichier

```
OVER   LI   R0,PAB
       LI   R1,ATTRIB
       LI   R2,2
       BLWP @VMBW
       LI   R9,PAB+9
       MOV  R9,@PNTR
       BLWP @DSRLNK
       DATA 8
       RT
```

* s/p d'ouverture

```
OPEN   LI   R0,PAB
       BLWP @VMBW
DSR    LI   R9,PAB+9
       MOV  R9,@PNTR
       BLWP @DSRLNK
       DATA 8
       JNE  RET2
       MOV  @INDX,@INDX
       JNE  RET3
       B    @ERREUR
RET2   RT
RET3   SETO @TERM
       RT
```

* s/p d'arret o/n

```
ARRET  BLWP @KSCAN
       MOVB @>837C,R4
       JEQ  ARRET
       MOVB @RTCLA,R4
       RT
```

* s/p de saisie

```
ACCEPT LI   R2,1
SG1    LI   R3,>100
SG2    BLWP @KSCAN
       MOVB @>837C,R4
       JEQ  SG16
       CB   @RTCLA,@ENT
       JEQ  SG12
       CB   @RTCLA,@BACK
       JNE  RED
       MOV  @DRAP6,@DRAP6
       JEQ  SG15
       MOV  R12,R12
       JNE  SG17
       B    @NOMGR
SG17   B    @NOUVE
SG15   MOV  @DRAP1,@DRAP1
       JEQ  SG13
       B    @OPT
SG13   MOV  @DRAP,@DRAP
       JEQ  SG2
       B    @IMPR
SG12   B    @FINI
SG16   B    @CURS
RED    CB   @RTCLA,@REDO
       JNE  SG7
       MOVB *R7,R1
       BLWP @VSBW
       MOV  @DRAP4,@DRAP4
       JNE  SG2
       MOV  @DRAP,@DRAP
       JEQ  SG5
       B    @SAISIE
SG5    MOV  @DRAP2,@DRAP2
       JEQ  SG6
       B    @AJGR2
SG6    MOV  @DRAP1,@DRAP1
       JEQ  SG7
       B    @AJOUT
SG7    CB   @RTCLA,@FCTNS
       JNE  SG3
       CI   R2,1
       JEQ  SG2
       MOVB *R7,R1
       BLWP @VSBW
       DEC  R2
       DEC  R7
       DEC  R0
       JMP  SG2
SG3    CB   @RTCLA,@FCTND
       JNE  SG4
       MOV  @DRAP2,@DRAP2
       JNE  SG2
       C    R2,@LSAIS
       JEQ  SG2
       MOVB *R7,R1
       BLWP @VSBW
       C    R2,@LSAIS
       JHE  SG11
       INC  R0
       INC  R2
       INC  R7
       JMP  SG2
SG4    MOVB @RTCLA,R1
       MOV  R12,R12
       JNE  SG18
       MOV  @DRAP2,@DRAP2
       JEQ  SG8
SG18   MOV  @DRAP6,@DRAP6
       JNE  SG10
       CB   R1,@OUI
       JEQ  SG8
       CB   R1,@NON
       JNE  SG2
       JMP  SG8
SG10   CB   R1,@ZERO
       JL   SG2
       CB   R1,@NEUF
       JH   SG2
SG8    MOVB R1,*R7+
       BLWP @VSBW
       MOV  @DRAP2,@DRAP2
       JNE  FINI1
       C    R2,@LSAIS
       JHE  SG11
       INC  R0
       INC  R2
       JMP  SG14
SG11   DEC  R7
       JMP  SG14
CURS   DEC  R3
       JNE  SG14
       MOV  R6,R6
       JEQ  SGCL
       CLR  R6
       LI   R1,>1E00
       BLWP @VSBW
       JMP  SG9
SGCL   SETO R6
       MOVB *R7,R1
       BLWP @VSBW
       JMP  SG9
FINI1  DEC  R7
```

* apres enter

```
FINI    MOVB *R7,R1
        BLWP @VSBW
RET1    RT
SG9     B    @SG1
SG14    B    @SG2
* s/p de retour
RETOUR BL   @CLEAR
        LI   R0,>01E0
        BLWP @VWTR
        SWPB R0
        MOVB R0,@>83D4
        CLR  @>837C
        LWPI >83E0
        B    @>0070
* s/p d'attente <enter><back>
ENTER   BLWP @KSCAN
        MOVB @>837C,R13
        ANDI R13,>FF00
        LI   R14,>2000
        COC  R14,R13
        JNE  ENTER
        CB   @RTCLA,@ENT
        JEQ  ENTER1
        CB   @RTCLA,@BACK
        JNE  ENTER
        RT
ENTER1 SETO @DRAP2
        RT
* sauve attribut en cas d'erreur
SATTR   INC  R1
        MOVB *R1,@ATTRIB+1
        DEC  R1
        RT
* s/p affichage en decimal
DECI    MOV  R2,R0
B3      MOV  R4,R5
        CLR  R4
        DIV  @DIX,R4
        MOV  R5,R1
        AI   R1,>30
        SWPB R1
        BLWP @VSBW
        DEC  R0
        C    R0,R7
        JNE  B3
        RT
******************
FICHES LI   R0,>0744
        BLWP @VWTR
        LI   R0,>01F0
        BLWP @VWTR
        SWPB R0
        MOVB R0,@>83D4
* retablir la couleur
        LI   R0,>07F4
        BLWP @VWTR
        CLR  @>8374
FICH    BL   @CLEAR
        LWPI REG
* affichage 1er ecran
        LI   R2,TEXTE
        LI   R3,ADDRE
        BL   @AFFICH
* index? o/n
        MOV  @CHTRI,@CHTRI
        JEQ  CHOIX
        LI   R2,TEXTAL
        LI   R3,ADD14
        BL   @AFFICH
ARRO    BL   @ARRET
        CB   R4,@NON
        JEQ  CHOIX
        CB   R4,@OUI
        JNE  ARRO
        CLR  @CHTRI
        CLR  @INDX2
* choix de depart
CHOIX   BL   @CLS24
        BL   @ARRET
        CB   R4,@UN
        JEQ  CH1
        CB   R4,@DEUX
        JEQ  CH2
        CB   R4,@TROIS
        JEQ  CH3
        CB   R4,@QUATRE
        JEQ  CH4
        CB   R4,@CINQ
        JNE  CHOIX+4
        B    @RETOUR
CH1     B    @GERER
CH2     B    @IMPRES
CH3     B    @INIT
CH4     B    @TRIER
*************************
* PROGRAMME DE GESTION
*************************
```

```
TEXT1 TEXT -'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ '
      TEXT -'@                               @ '
      TEXT -'@ Nombre de fiches triees = 000 @ '
      TEXT -'@                               @ '
      TEXT -'@ Nombre total de fiches  = 000 @ '
      TEXT -'@                               @ '
      TEXT -'@ Nb de groupes utilises  = 000 @ '
      TEXT -'@                               @ '
      TEXT -'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ '
      TEXT -'Nom de l''imprimante = '
      TEXT -'<ENTER> pour continuer '
ADD1    DATA
284,324,364,404,444,484,524,564,604,686,849,0
TEXT2   TEXT -'XXXXXXXXXXXXXXXXXXXXXXX '
        TEXT -'X  FICHIER D''ADRESSES  X '
        TEXT -'XXXXXXXXXXXXXXXXXXXXXXX '
        TEXT -'GESTION DU FICHIER '
        TEXT -'~~~~~~~~~~~~~~~~~~ '
        TEXT -'1-Mise a jour '
        TEXT -'2-Groupes '
        TEXT -'3-Recherche '
        TEXT -'4-Retour '
ADD2    DATA 48,88,128,331,371,491,531,571,611,0
TEXT3   TEXT -'xxxxxxxxxxxxxxxxxxxxxxxxx '
        TEXT -'x mise a jour du fichier x '
        TEXT -'xxxxxxxxxxxxxxxxxxxxxxxxx '
        TEXT -'OPTIONS DE MISE A JOUR '
        TEXT -'~~~~~~~~~~~~~~~~~~~~~~ '
        TEXT -'1-Ajouter '
        TEXT -'2-Modifier '
        TEXT -'3-Effacer '
        TEXT -'4-Retour '
ADD3    DATA 7,47,87,329,369,494,534,574,614,0
TEXT4   TEXT -'xxxxxxxxxxxxxxxxxxxxxxxxxx '
        TEXT -'x modification de groupe x '
        TEXT -'xxxxxxxxxxxxxxxxxxxxxxxxxx '
        TEXT -'OPTIONS '
        TEXT -'~~~~~~~ '
        TEXT -'1-Ajouter un groupe '
        TEXT -'2-Corriger un groupe '
        TEXT -'3-Effacer un groupe '
        TEXT -'4-Fiches hors groupes '
        TEXT -'5-Retour '
AD4     DATA 7,47,87,335,375,450,490,530
        DATA 570,610,0
TEXT5   TEXT -'xxxxxxxxxxxxxxxxxxxxxxxxxx '
        TEXT -'x recherche dans fichier x '
        TEXT -'xxxxxxxxxxxxxxxxxxxxxxxxxx '
        TEXT -'OPTIONS DE RECHERCHE '
        TEXT -'~~~~~~~~~~~~~~~~~~~~ '
        TEXT -'1-Une fiche '
        TEXT -'2-Un groupe '
        TEXT -'3-Tout le fichier '
        TEXT -'4-Retour '
ADD5    DATA 7,47,87,330,370,492,532,572,612,0
TEXT6   TEXT -'FICHE No '
        TEXT -'TITRE: '
        TEXT -'NOM Prenom: '
        TEXT -'ADRESSE: '
        TEXT -'CODE PTT: '
```

```
       TEXT -'VILLE: '
       TEXT -'TELEPHONE: '
       TEXT -'COTISATION: '
       TEXT -'No LIC: '
       TEXT -'BOX: '
ADD6   DATA 187,205,280,363,442
       DATA 525,601,680,764,847,0
TEXT7  TEXT 'Valider : <ENTER>  Corriger : '
       TEXT -'<BACK> '
ADD7   DATA 922,0
TEXT8  TEXT -'Une autre fiche ?  O/N '
ADD8   DATA 927,0
TEXT9  TEXT 'indiquez par O/N le groupe dans '
       TEXT -'lequel '
       TEXT 'vous voulez mettre le nouveau '
       TEXT -'nom '
ADD9   DATA 280,322,0
TEXT10 TEXT 'LES 450 FICHES PREVUES SONT '
       TEXT -'UTILISEES '
       TEXT -'Initialisez un autre fichier '
       TEXT -'sur une autre disquette '
       TEXT -'Appuyez sur une touche '
ADD10  DATA 401,524,566,847,0
       EVEN
```

```
* s/p evite repetition touche
BLOQ   MOVB @>837C,R13
       ANDI R13,>FF00
       LI   R14,>2000
       COC  R14,R13
       RT
* s/p d'ecriture
WRITE  LI   R0,TPAB
       LI   R1,GROUPE
       LI   R2,94
       BLWP @VMBW
       AI   R0,94
       LI   R1,NOMIMP
       LI   R2,33
       BLWP @VMBW
       RT
***************************
* ouvre le fichier "index"
GERER  MOV  @INDX2,@INDX2
       JNE  DON1
       LI   R1,DSK2
       BL   @SATTR
       LI   R2,20
       BL   @OPEN
       SETO @INDX
       CLR  R5
* lecture des enregistrements
INDEX1 BL   @INPUT
       BL   @DSR
       MOV  @TERM,@TERM
       JEQ  TRANSF
* verifier si err=5 (fin fichier)
       CLR  @TERM
       LI   R0,PAB+1
       BLWP @VSBR
       SRL  R1,13
       CI   R1,5
       JEQ  FERME
       B    @ERREUR
* fermeture "index"
FERME  BL   @OVER
       CLR  @INDX
       SETO @INDX2
       JMP  DON1
* transfer dans "index"
TRANSF LI   R0,TPAB
       LI   R1,INDEX
       A    R5,R1
       LI   R2,254
       BLWP @VMBR
       AI   R5,254
       JMP  INDEX1
* ouverture fichier "donnees"
DON1   MOV  @DRAP10,@DRAP10
       JEQ  DON2
       B    @IMPR1
DON2   LI   R1,DSK1
       BL   @SATTR
       LI   R2,22
       BL   @OPEN
* lecture enregistrement 0
       BL   @INPUT
       BL   @DSR
* saisie groupes,tri,tot,ngr,imp
       BL   @READ
* ecriture a l'ecran
ECRAN1 BL   @CLEAR1
       LI   R2,TEXT1
       LI   R3,ADD1
       BL   @AFFICH
       LI   R2,394
       LI   R7,391
       MOV  @TRI,R4
       BL   @DECI
       AI   R2,80
       AI   R7,80
       MOV  @TOT,R4
       BL   @DECI
       AI   R2,80
       AI   R7,80
       MOV  @NGR,R4
       BL   @DECI
       LI   R0,708
       LI   R1,NOMIMP+1
       LI   R2,32
       BLWP @VMBW
       BL   @ENTER
* ecran suivant
OPT    BL   @CLEAR
       LI   R2,TEXT2
       LI   R3,ADD2
       BL   @AFFICH
       CLR  R12
       CLR  @DRAP
       CLR  @DRAP1
       CLR  @DRAP2
       CLR  @DRAP3
       CLR  @DRAP4
       CLR  @DRAP5
       CLR  @DRAP6
       CLR  @DRAP7
       CLR  @DRAP8
       CLR  @DRAP9
       CLR  @DRAP10
       CLR  @DRAP11
* choix
CHOIX1 BLWP @ARRET
       BL   @BLOQ
       JNE  CHOIX1
       CB   R4,@UN
       JNE  CH7
       JMP  JOUR
CH7    CB   R4,@DEUX
       JNE  CH8
       B    @NOMGR
CH8    CB   R4,@TROIS
       JNE  CH9
       B    @RECHER
CH9    CB   R4,@QUATRE
       JNE  CHOIX1
* ferme le fichier
       BL   @OVER
       B    @FICH
* options mise a jour
JOUR   BL   @CLEAR
       LI   R2,TEXT3
       LI   R3,ADD3
       BL   @AFFICH
CHOIX2 BLWP @ARRET
       BL   @BLOQ
       JNE  CHOIX2
       CB   R4,@UN
       JNE  CH10
       B    @AJOUTE
CH10   CB   R4,@DEUX
       JNE  CH11
       B    @MODIF
CH11   CB   R4,@TROIS
       JNE  CH12
       B    @EFFACE
CH12   CB   R4,@QUATRE
       JNE  CHOIX2
       B    @OPT
* options groupes
NOMGR  BL   @CLEAR
       CLR  @DRAP6
       LI   R2,TEXT4
       LI   R3,AD4
       BL   @AFFICH
CHOIX3 BLWP @ARRET
```

```
       BL    @BLOQ
       JNE   CHOIX3
       CB    R4,@UN
       JL    CHOIX3
       CB    R4,@TROIS
       JH    CH14
       B     @MODGR
CH14   CB    R4,@QUATRE
       JNE   CH18
       B     @APPGR
CH18   CB    R4,@CINQ
       JNE   CHOIX3
       B     @OPT
* options de recherche
RECHER BL    @CLEAR
       LI    R2,TEXT5
       LI    R3,ADD5
       BL    @AFFICH
CHOIX4 BLWP  @ARRET
       BL    @BLOQ
       JNE   CHOIX4
       CB    R4,@UN
       JNE   CH15
       B     @RECH1
CH15   CB    R4,@DEUX
       JNE   CH16
       B     @AFFIGR
CH16   CB    R4,@TROIS
       JNE   CH17
       B     @TOUTFI
CH17   CB    R4,@QUATRE
       JNE   CHOIX4
       B     @OPT
* s/p efface la derniere ligne
CLS24  LI    R1,NET
       LI    R0,920
       LI    R2,40
       BLWP  @VMBW
       RT
******************
* AJOUTER UN NOM
******************
AJOUTE BL    @CLEAR1
* test si tot<=450
       C     @TOT,@D450
       JLE   AJOUT2
       LI    R2,TEXT10
       LI    R3,ADD10
       BL    @AFFICH
       BL    @OVER
       BL    @ARRET
       B     @FICH
* test positif
AJOUT2 LI    R1,>2E00
       BL    @INITAM
       LI    R2,TEXT6
       LI    R3,ADD6
       BL    @AFFICH
       MOV   @TOT,R4
       INC   R4
* No de fiche
AJOUT4 LI    R2,198
       LI    R7,195
       BL    @DECI
AJOUT  SETO  @DRAP1
       CLR   @DRAP
       CLR   @NLIG
       CLR   @LGR
       LI    R8,NBSAIS
* saisie des informations
AJOUT1 LI    R0,212
       A     @NLIG,R0
       CLR   @DRAP2
       MOV   R0,R13   ----> adr ecran
       MOV   *R8+,@LSAIS    depart
       JEQ   AJFIN
       LI    R1,TAMPON
       A     @LGR,R1
       MOV   R1,R7
       MOV   @LSAIS,R2
       BLWP  @VMBW
       BL    @ACCEPT
* verifie si appel depuis "modif"
       MOV   @DRAP4,@DRAP4
       JNE   AJOUT3
* recherche dans fichier du nom
       CI    R13,292
       JNE   AJOUT3
       MOV   @TOT,@TOT
       JEQ   AJOUT3
       LI    R0,TPAB+4
       LI    R1,TAMPON+4
       LI    R2,28
       BLWP  @VMBW
       BL    @ADDIT
       MOV   @ASCII,@SOMME1
       SETO  @DRAP3
       BL    @RBLOC
       MOV   @DRAP3,@DRAP3
       JNE   AJOUT3
       LI    R8,NBSAIS+2
       BL    @CLS24
       JMP   AJOUT1
* lignes suivantes
AJOUT3 A     @D80,@NLIG
* LGR=nb a ajouter a "tampon"
       A     @LSAIS,@LGR
       JMP   AJOUT1
* affiche <enter><back>
AJFIN  LI    R2,TEXT7
       LI    R3,ADD7
       BL    @AFFICH
       CLR   @DRAP3
       BL    @ENTER
       MOV   @DRAP2,@DRAP2
       JNE   AJFIN1
       BL    @CLS24
       B     @AJOUT
* poursuité de l'operation
AJFIN1 MOV   @DRAP4,@DRAP4
       JNE   AJFIN2
       INC   @TOT
AJFIN2 BL    @CLEAR1
       CLR   @LSAIS
       INC   @LSAIS
* affiche le nom
       LI    R0,165
       LI    R3,TAMPON
       LI    R2,32
AJG1   MOVB  *R3+,R1
       CB    R1,@POINT
       JEQ   AJG2
       BLWP  @VSBW
AJG2   INC   R0
       DEC   R2
       JNE   AJG1
* appartenance a un groupe
       LI    R2,TEXT9
       LI    R3,ADD9
       BL    @AFFICH
       LI    R2,TEXTG1
       LI    R3,ADGR+4
       BL    @AFFICH
AJGR   MOV   @DRAP4,@DRAP4
       JNE   AJGR2
       LI    R2,TEXTG3
       LI    R3,ADG3
       BL    @AFFICH
       LI    R3,9
       LI    R2,BUFGR
       LI    R1,>2000
BUFCL  MOVB  R1,*R2+
       DEC   R3
       JNE   BUFCL
AJGR2  CLR   @DRAP1
       CLR   @NLIG
       CLR   @LGR
* affichage du groupe
AJGR1  LI    R0,486
       A     @NLIG,R0
       LI    R1,GROUPE
       A     @LGR,R1
       LI    R2,11
       BLWP  @VMBW
       INC   @DRAP1
* saisie O ou N
ON     LI    R0,498
       A     @NLIG,R0
       LI    R1,BUFGR+1
       A     @DRAP1,R1
       DEC   R1
       BLWP  @VSBW
       MOV   R1,R7
       BL    @ACCEPT
* groupe suivant
       C     @DRAP1,@NGR
       JEQ   ENTBAC
       A     @D40,@NLIG
       A     @D11,@LGR
       JMP   AJGR1
ENTBAC CLR   @DRAP1
* enter ou back
       LI    R2,TEXT7
       LI    R3,ADD7
       BL    @AFFICH
       CLR   @DRAP2
       BL    @ENTER
       MOV   @DRAP2,@DRAP2
       JNE   GRTRAN
       BL    @CLS24
       JMP   AJGR2
* transfer et verif. si inactif
```

```
GRTRAN CLR   R3
       LI    R5,118
       CLR   @DRAP1
       LI    R1,BUFGR
       LI    R2,TAMPON+119
GRTR   INC   R1
       INC   R3
       C     R3,@NGR
       JH    AJSUIT
       MOVB  *R1,*R2+
       CB    *R1,@OUI
       JNE   GRTR
       SETO  @DRAP1
       JMP   GRTR
AJSUIT MOV   @DRAP1,@DRAP1
       JEQ   INACT
* marque actif
       MOVB  @A,@TAMPON(R5)
       JMP   AJENR
* marque inactif
INACT  MOVB  @I,@TAMPON(R5)
AJENR  CLR   @DRAP1
* transfert dans tpab
       LI    R0,TPAB
       LI    R1,TAMPON
       LI    R2,127
       BLWP  @VMBW
* No de l'enregistrement
AJENR4 LI    R0,PAB+6
       MOV   @DRAP4,@DRAP4
       JEQ   AJENR1
       LI    R1,NUMENR
       JMP   AJENR2
AJENR1 LI    R1,TOT
AJENR2 LI    R2,2
       BLWP  @VMBW
* mode update
       LI    R0,PAB+1
       LI    R1,>0900
       BLWP  @VSBW
* enregistrement de la fiche
       BL    @PRINT
       BL    @DSR
       MOV   @DRAP5,@DRAP5
       JNE   AJ2
       MOV   @DRAP4,@DRAP4
       JEQ   AJENR3
       B     @AUTMOD
* autre fiche ?
AJENR3 BL    @CLS24
       LI    R2,TEXT8
       LI    R3,ADD8
       BL    @AFFICH
ARR    BL    @ARRET
       CB    R4,@OUI
       JEQ   AJ1
       CB    R4,@NON
       JNE   ARR
       CLR   @DRAP2
       JMP   ENREG0
AJ1    B     @AJOUTE
AJ2    B     @AUTEFF
* enregistrer tri,tot,ngr...
ENREG0 CLR   @NUMENR
       BL    @NUM
       BL    @WRITE
       LI    R0,PAB+1
       LI    R1,>0900
       BLWP  @VSBW
       BL    @PRINT
       BL    @DSR
       B     @OPT
*************************************
*  Dans chaque enregistrement
*  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* ENR+0   = titre        (4 octets)
* ENR+4   = nom prenom(28 octets)
* ENR+32  = adresse     (28 octets)
* ENR+60  = code ptt    (5 octets
* ENR+65  = ville        (20 octets)
* ENR+85  = telephone (15 octets)
* ENR+100= cotisation (4 octets)
* ENR+104= no de lic   (6 octets)
* ENR+110= box          (8 octets)
* ENR+118= act,ina,eff(1 octet)
* ENR+119= app.groupe (8 octets)
*               --------
*         TOTAL: 127 octets
*************************************
```

Source "FICH2"

```
*************************************
*  PROGRAMME D'INITIALISATION
*************************************
TITRE4 TEXT -'INITIALISER LE FICHIER '
       TEXT -'~~~~~~~~~~~~~~~~~~~~~~ '
       TEXT 'Chargez la disquette sur le '
       TEXT -'lecteur   No 1 '
       TEXT -'<ENTER> ou <BACK> '
ADDRE4 DATA 329,369,520,769,0
TITR4  TEXT 'Le nb de secteurs libres est '
       TEXT -'insuffisant '
       TEXT 'Le fichier complet occupera 227 '
       TEXT -'secteurs '
       TEXT 'Changez votre disquette : '
       TEXT -'<ENTER> '
ADDR4  DATA   520,560,763,0
TIT4   TEXT 'Le fichier complet occupera 227 '
       TEXT -'secteurs '
       TEXT -'Pour les 450 fiches prevues. '
       TEXT -'Pour ouvrir le fichier : <ENTER> '
ADD4   DATA 520,565,764,0
TEXTGR TEXT -'SAISIE DES NOMS DE GROUPES '
       TEXT -'~~~~~~~~~~~~~~~~~~~~~~~~~~ '
TEXTG1 TEXT -'******************* '
       TEXT -'*                 * '
       TEXT -'* 1-              * '
       TEXT -'* 2-              * '
       TEXT -'* 3-              * '
       TEXT -'* 4-              * '
       TEXT -'* 5-              * '
       TEXT -'* 6-              * '
       TEXT -'* 7-              * '
       TEXT -'* 8-              * '
       TEXT -'*                 * '
       TEXT -'******************* '
TEXTG2 TEXT -'<BACK> terminer '
TEXTG3 TEXT -'<REDO> corriger '
       TEXT -'<S><D> deplacer '
       TEXT -'<ENTER> valider '
ADG3   DATA 623,0
ADGR2  DATA 463,583,823,703,0
ADGR   DATA 287,327,402,442,482,522,562,602,642
       DATA 682,722,762,802,842,0
TEXT0  TEXT -'NOM DU DISPOSITIF D''IMPRESSION '
       TEXT -'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ '
ADD0   DATA 404,444,0
NBSAIS DATA 4,28,28,5,20,15,4,6,8,0
ALPHA  DATA 0
       EVEN
INIT   BL    @CLEAR1
       LI    R2,TITRE4
       LI    R3,ADDRE4
       BL    @AFFICH
* enter ou back
       BL    @ENTER
       CB    @RTCLA,@BACK
       JNE   INIT1
       B     @FICH
* ouverture catalogue disquette
INIT1  LI    R1,DSK0
       BL    @SATTR
       LI    R2,15
```

```
        BL    @DSR
* lecture du catalogue rec:0
        BL    @INPUT
        BL    @DSR
* fermeture
        BL    @OVER
* transfert rec 0 dans tampon
        LI    R0,TPAB
        LI    R1,TAMPON
        LI    R2,38
        BLWP  @VMBR
* longueur du nom dans R3
        MOVB  *R1,R3
        SRL   R3,8
* +20 pour secteurs restants
        AI    R3,20
* transfert des 8 octets dans FAC >834A
        LI    R0,TPAB
        A     R3,R0
        LI    R1,>834A
        LI    R2,8
        BLWP  @VMBR
* tranformer en nombre simple
        BLWP  @XMLLNK
        DATA  >1200
        MOV   @>834A,@SECT
* comparer avec le minimum
        C     @SECT,@MINI
        JHE   OK
* si pas assez de secteurs libres
        LI    R2,TITR4
        LI    R3,ADDR4
        BL    @AFFICH
        BL    @ARRET
        B     @INIT
* nb suffisant
OK      LI    R2,TIT4
        LI    R3,ADD4
        BL    @AFFICH
        BL    @ARRET
* ouverture fichier donnees
        LI    R0,PAB
        LI    R1,DSK1
        BL    @SATTR
        LI    R2,22
        BLWP  @VMBW
* mode update
        LI    R0,PAB+1
        LI    R1,>0900
        BLWP  @VSBW
        BL    @DSR
* saisie nom de groupe
SAISIE  BL    @CLEAR1
        CLR   @NGR
        CLR   @NLIG
        CLR   @LSAIS
        SETO  @DRAP
        CLR   @DRAP2
        CLR   @LGR
        LI    R2,TEXTGR
        LI    R3,ADGR
        BL    @AFFICH
        LI    R2,TEXTG2
        LI    R3,ADGR2
        BL    @AFFICH
        LI    R1,>2000
        BL    @INITAM
*********************************
* dans l'enregistrement No 0 :
* de TAMPON a TAMPON+87 il y a
* tous les 11 octets les noms
* des 8 groupes possibles.
* TRI est a TAMPON+88(2 octets)
* TOT est a TAMPON+90(2 octets)
* NGR est a TAMPON+92(2 octets)
* TAMPON+94=long. nom imprim.
* TAMPON+95=nom imp (32 octets)
*********************************
* saisie des groupes
SG0     LI    R0,486
        A     @NLIG,R0
        MOV   R0,R8
        LI    R1,GROUPE
        A     @LGR,R1
        MOV   R1,R7
        LI    R2,11
        BLWP  @VMBW
        MOV   R2,@LSAIS
        BL    @ACCEPT
* verifie si ce n'est pas un blanc
        MOV   R8,R0
        CLR   R1
        BLWP  @VSBR
        CI    R1,>2000
        JEQ   SG0
* compteur de groupe
        INC   @NGR
* saisie des lignes suivantes
        A     @D40,@NLIG
        A     @D11,@LGR
        C     @NLIG,@D320
        JNE   SG0
        CLR   @DRAP10
* saisie nom imprimante
IMPR    BL    @CLEAR1
        LI    R2,TEXTO
        LI    R3,ADD0
        BL    @AFFICH
        LI    R0,524
        LI    R1,NOMIMP+1
        LI    R2,32
        BLWP  @VMBW
        MOV   R1,R7
        MOV   R2,@LSAIS
        CLR   @DRAP
        CLR   @DRAP2
        BL    @ACCEPT
        DEC   R2
        SWPB  R2
        MOVB  R2,@NOMIMP
        MOV   @DRAP10,@DRAP10
        JEQ   IM
* Mode update rec 0
        BL    @CLEAR1
        LI    R0,PAB+1
        LI    R1,>0900
        BLWP  @VSBW
        CLR   @NUMENR
        BL    @NUM
        JMP   IM1
* met le fichier a zero
IM      CLR   @TRI
        CLR   @TOT
* transfert des saisies
        LI    R0,TPAB
        LI    R1,GROUPE
        LI    R2,94
        BLWP  @VMBW
IM1     LI    R0,TPAB+94
        LI    R1,NOMIMP
        LI    R2,33
        BLWP  @VMBW
* enregistrement
        BL    @PRINT
        BL    @DSR
        BL    @OVER
        MOV   @DRAP10,@DRAP10
        JEQ   IM2
        BL    @DELAI
        B     @IMPR1
* initialisation de "index"
IM2     LI    R1,DSK2
        BL    @SATTR
        LI    R2,20
        LI    R0,PAB
        BLWP  @VMBW
* mode output
        LI    R0,PAB+1
        LI    R1,>0A00
        BLWP  @VSBW
        BL    @DSR
* enregistrement 4 octets nuls
        LI    R0,TPAB
        LI    R1,TRI
        LI    R2,4
        BLWP  @VMBW
        BL    @PRINT
        BL    @DSR
* fermeture fichier
FERM    BL    @OVER
        B     @FICH
X       DATA  0
Y       DATA  0
XY      DATA  0
ASCII   DATA  0
SOMME1  DATA  0
SOMME2  DATA  0
SAVRET  DATA  0
DRAP3   DATA  0
NUMENR  DATA  0
BLANC   BYTE  32
POINT   BYTE  46
        EVEN
* s/p d'ecriture sur dsk
PRINT   LI    R0,PAB
        LI    R1,>0300
        BLWP  @VSBW
        RT
* s/p de lecture sur dsk
INPUT   LI    R0,PAB
        LI    R1,>0200
        BLWP  @VSBW
```

```
        RT
* s/p d'addition ascii
ADDIT   CLR  @ASCII
        CLR  R1
        BLWP @VSBR
* on x le 1er octet par 512
* pour qu'il soit significatif
* (on decale l'octet fort de R1 a gauche)
        SLA  R1,1
        MOV  R1,R13
        INC  R0
        BLWP @VSBR
* on x le 2ieme octet par 16
        SRL  R1,4
        A    R1,R13
        CLR  R14
* plus l' octet suivant
T1      INC  R0
        BLWP @VSBR
* cas d'un espace
        CB   R1,@BLANC
        JNE  T2
        MOV  R0,R14
* cas d'un point
T2      CB   R1,@POINT
        JEQ  T5
        SRL  R1,8
        A    R1,R13
        MOV  R14,R14
        JNE  T4
* chercher l'espace pour ajouter
* eventuellement le 1er octet prenom
* sauf si mode tri temporaire
        MOV  @CHTRI,@CHTRI
        JEQ  T6
        LI   R2,5
        JMP  T3
T6      LI   R2,15
T3      INC  R0
        DEC  R2
        JEQ  T5
        BLWP @VSBR
        CB   R1,@BLANC
        JEQ  T4
        CB   R1,@POINT
        JEQ  T5
        JMP  T3
* 1er octet du prenom
T4      INC  R0
        BLWP @VSBR
        SRL  R1,8
        A    R1,R13
T5      MOV  R13,@ASCII
        RT
* s/p de recherche en bloc
RBLOC   MOV  R11,@SAVRET
        CLR  @SOMME2
        MOV  @TOT,@TOT
        JEQ  RS2
        MOV  @TRI,@TRI
        JEQ  RSEQ
        CLR  @X
        INC  @X
        MOV  @TRI,@Y
* coupe le bloc en deux
DIVISE  CLR  @XY
        A    @X,@XY
        A    @Y,@XY
        MOV  @XY,R2
        SRL  R2,1
        MOV  R2,@XY
        JEQ  RSEQ
* passer par "index"
        MOV  R2,@ALPHA
        DEC  R2
        SLA  R2,1
        MOV  @INDEX(R2),@NUMENR
        MOV  @NUMENR,@SECT
* aller chercher l'enregistrement
        BL   @NUM
* lecture et addition ascii
        BL   @INPUT
        BL   @DSR
        LI   R0,TPAB+4
        BL   @ADDIT
        C    @SOMME2,@ASCII
        JEQ  RSEQ
        MOV  @ASCII,@SOMME2
* comparaison avec le nom recherche
        C    @ASCII,@SOMME1
        JEQ  RS6
        C    @X,@Y
        JEQ  RSEQ
        C    @ASCII,@SOMME1
        JL   PART2
* si plus grand que nom recherche
        DEC  @XY
        MOV  @XY,@Y
        JMP  DIVISE
* si plus petit que nom recherche
PART2   INC  @XY
        MOV  @XY,@X
        JMP  DIVISE
* s/p de recherche sequentielle
RSEQ    C    @TOT,@TRI
        JEQ  RS3
        MOV  @TRI,@NUMENR
RS1     INC  @NUMENR
        C    @NUMENR,@TOT
        JH   RS3
        MOV  @NUMENR,@ALPHA
        MOV  @NUMENR,@SECT
* no de l'enregistrement
        LI   R0,PAB+6
        LI   R1,NUMENR
        LI   R2,2
        BLWP @VMBW
* lecture+addition ascii
        BL   @INPUT
        BL   @DSR
        LI   R0,TPAB+4
        BL   @ADDIT
* verifie si c'est le bon nom
        C    @SOMME1,@ASCII
        JNE  RS1
RS6     MOV  @DRAP3,@DRAP3
        JNE  RS5
        JMP  RS4
* le fichier est vide
RS2     LI   R2,TEXT13
        LI   R3,ADD14
        BL   @AFFICH
        BL   @ENTER
        SETO @DRAP
        JMP  RS4
* pas trouve le nom
RS3     MOV  @DRAP3,@DRAP3
        JNE  RS4
        LI   R2,TEXT12
        LI   R3,ADD14
        BL   @AFFICH
        BL   @ENTER
        SETO @DRAP
RS4     MOV  @SAVRET,R11
        RT
* nom deja sur fichier
RS5     LI   R2,TEXT14
        LI   R3,ADD14
        BL   @AFFICH
        BL   @ENTER
        CLR  @DRAP3
        JMP  RS4
*********************
* PROGRAMME DE TRI
*********************
TEXTE2 TEXT -'* ERREUR    * '
```

```
        TEXT -'<ENTER> pour continuer '
ERR1    TEXT -'   Protection contre l''ecriture '
ERR2    TEXT 'Le fichier n''est pas sur la '
        TEXT -'disquette '
ERR5    TEXT -'    Lecture au-dela du fichier '
ERR6    TEXT 'La disquette n''est pas sur le '
        TEXT -'lecteur '
ERRG    TEXT ' Sur l''entree/sortie du '
        TEXT -'peripherique '
ADER1   DATA 332,567,0
ADER2   DATA 441,0
TEX1    TEXT -'XXXXXXXXXXXXXXXXXXX '
        TEXT -'X PROGRAMME DE TRI X '
        TEXT -'XXXXXXXXXXXXXXXXXXX '
        TEXT -'OPTIONS DE TRI '
        TEXT -'~~~~~~~~~~~~~~ '
        TEXT -'1-Tri alphabetique permanent '
        TEXT -'2-Tri temporaire au choix '
        TEXT -'3-Retour '
ADTEX1  DATA 48,88,128,251,291,365,405,445,0
TEX2    TEXT -'TRI TEMPORAIRE '
        TEXT -'~~~~~~~~~~~~~~ '
```

```
       TEXT -'Choix   pour trier '
       TEXT -'1       code ptt '
       TEXT -'2       ville '
       TEXT -'3       cotisation '
       TEXT -'4       box '
ADTEX2 DATA 211,251,329,411,451,491,531,571,0
TEXT27 TEXT -'TRI ALPHABETIQUE DES 000 FICHES '
       TEXT -'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ '
       TEXT -'1-Aucune suppression '
       TEXT -'2-Supprimer les effacements '
       TEXT -'3-effacements et inactifs '
       TEXT -'4-Retour '
ADD27  DATA 243,283,365,405,445,485,0
TEXT28 TEXT '########## POUR LE TRI ##########'
       TEXT -'# '
       TEXT '#                                '
       TEXT -'# '
       TEXT '# Le Nb de fiches va de 1 a 000  '
       TEXT -'# '
       TEXT '#                                 '
       TEXT -'# '
       TEXT '##################################'
       TEXT -'# '
TEXTOP TEXT ' ======= OPERATION EN COURS '
       TEXT -'======= '
ADD28  DATA 602,642,682,722,762,881,0
TEXT29 TEXT 'OPERATION DE TRI TERMINEE...'
       TEXT -'<ENTER> '
ADD29  DATA 882,0
TEXTAL TEXT -'REVENIR TRI ALPHABETIQUE? O/N '
TEXT46 TEXT 'CLASSEMENT "INACTIF" TOUTES '
       TEXT -'FICHES '
       TEXT ' ETES-VOUS SUR DE VOTRE CHOIX ? '
       TEXT -'O/N '
ADD46  DATA 603,682,0
GROFF  BYTE 73,78,78,78,78,78,78,78,78
```

```
ERR    BYTE 0
       EVEN
CHTRI  DATA 0
INVER1 DATA 0
INVER2 DATA 0
K      DATA 0
M      DATA 0
ECART  DATA 0
INDEX  BSS  900
STOCK  BSS  1800
* s/p de numerotation
NUM    LI   R1,NUM
       LI   R0,PAB+6
       LI   R2,2
       BLWP @VMBW
       RT
* s/p traitement erreur
ERREUR BL   @CLEAR
       LI   R0,PAB+1
       BLWP @VSBR
       ANDI R1,>E000
       SRL  R1,5
       MOVB R1,@ERR
* fermeture fichier
       LI   R0,PAB
       LI   R1,ATTRIB
       LI   R2,2
       BLWP @VMBW
       LI   R9,PAB+9
       MOV  R9,@PNTR
       BLWP @DSRLNK
       DATA 8
* no d'erreur
       MOVB @ERR,R1
       SRL  R1,8
       MOV  R1,R5
       CI   R5,2
       JNE  RR1
       LI   R2,ERR2
       JMP  ERR3
RR1    CI   R5,1
       JNE  RR5
       LI   R2,ERR1
       JMP  ERR3
RR5    CI   R5,5
       JNE  RR6
       LI   R2,ERR5
       JMP  ERR3
RR6    CI   R5,6
       JNE  RRG
       LI   R2,ERR6
       JMP  ERR3
* affiche erreur
RRG    LI   R2,ERRG
       LI   R3,ADER2
       BL   @AFFICH
       JMP  ERR4
ERR3   LI   R3,ADER2
       BL   @AFFICH
ERR4   LI   R2,TEXTE2
       LI   R3,ADER1
       BL   @AFFICH
       LI   R0,341
       MOV  R5,R1
       AI   R1,48
       SWPB R1
       BLWP @VSBW
BKSCAN BLWP @KSCAN
       CB   @RTCLA,@ENT
       JNE  BKSCAN
       B    @FICH
* ouverture fichier "donnees"
TRIER  LI   R1,DSK1
       BL   @SATTR
       LI   R2,22
       BL   @OPEN
* enregistrement 0
       BL   @INPUT
       BL   @DSR
       BL   @READ
* mise en mode update
       LI   R0,PAB+1
       LI   R1,>0900
       BLWP @VSBW
* options
OPT1   BL   @CLEAR
       LI   R2,TEX1
       LI   R3,ADTEX1
       BL   @AFFICH
ARR11  BL   @ARRET
       CB   R4,@UN
       JEQ  TRIER1
       CB   R4,@DEUX
       JEQ  OPT2
       CB   R4,@TROIS
       JNE  ARR11
       BL   @OVER
       B    @FICH
* option 2
OPT2   BL   @CLEAR1
       SETO @CHTRI
       LI   R2,TEX2
       LI   R3,ADTEX2
       BL   @AFFICH
ARR12  BL   @ARRET
       CB   R4,@UN
       JNE  C2
       LI   R8,TPAB+60
       JMP  TRIER2
C2     CB   R4,@DEUX
       JNE  C3
       LI   R8,TPAB+65
       JMP  TRIER2
C3     CB   R4,@TROIS
       JNE  C4
       LI   R8,TPAB+100
       JMP  TRIER2
C4     CB   R4,@QUATRE
       JNE  ARR12
       LI   R8,TPAB+110
TRIER2 CLR  @NUMENR
       SETO @INDX2
       LI   R10,STOCK
* saisie des informations a trier
TRI8   INC  @NUMENR
       C    @NUMENR,@TOT
       JH   TRIER3
       BL   @NUM
       BL   @INPUT
       BL   @DSR
* info en somme code ascii
       MOV  R8,R0
```

```
       BL    @ADDIT
* trans. no + ascii dans "stock"
       MOV   @NUMENR,*R10+
       MOV   @ASCII,*R10+
       JMP   TRI8
TRIER3 B     @TRIER4
* option 1
TRIER1 BL    @CLEAR1
       CLR   @CHTRI
       LI    R2,TEXT27
       LI    R3,ADD27
       BL    @AFFICH
       MOV   @TOT,R4
       LI    R2,266
       LI    R7,263
       BL    @DECI
ARR6   BL    @ARRET
       CB    R4,@UN
       JEQ   TRI0
       CB    R4,@DEUX
       JEQ   TRI0
       CB    R4,@TROIS
       JEQ   TRI0
       CB    R4,@QUATRE
       JNE   ARR6
       B     @OPT1
TRI0   CLR   @NUMENR
       CLR   @INDX2
       LI    R10,STOCK
* saisie des noms
TRI1   INC   @NUMENR
       C     @NUMENR,@TOT
       JH    TRI5
       BL    @NUM
       BL    @INPUT
       BL    @DSR
* options
       CB    R4,@UN
       JEQ   TRI4
       LI    R0,TPAB+118
       BLWP  @VSBR
       CB    R4,@TROIS
       JNE   TRI6
       CB    R1,@I
       JEQ   TRI2
TRI6   CB    R1,@E
       JNE   TRI4
TRI7   C     @NUMENR,@TOT
       JNE   TRI2
       DEC   @TOT
       JMP   TRI1
* dernier enregis. a la place de
* celui qui est marque "e" ou "i"
TRI2   LI    R0,PAB+6
       LI    R1,TOT
       LI    R2,2
       BLWP  @VMBW
       BL    @DSR
* verif. si "e" ou "i" lui aussi
       LI    R0,TPAB+118
       BLWP  @VSBR
       CB    R1,@E
       JEQ   TRI3
       CB    R1,@I
```

```
       JEQ   TRI3
* prends la place du "e" ou "i"
       BL    @NUM
       BL    @PRINT
       BL    @DSR
       DEC   @TOT
       JMP   TRI4
TRI3   DEC   @TOT
       JMP   TRI7
* nom en somme code ascii
TRI4   LI    R0,TPAB+4
       BL    @ADDIT
* trans. no + ascii dans "stock"
       MOV   @NUMENR,*R10+
       MOV   @ASCII,*R10+
       JMP   TRI1
* enregitrement de rec 0
TRI5   MOV   @TOT,@TRI
       BL    @WRITE
       CLR   @NUMENR
       BL    @NUM
       BL    @PRINT
       BL    @DSR
* fermeture du fichier "donnees"
TRIER4 BL    @OVER
***********************************
* chaque enreg. valide est dans
* stock sur 4 octets.
* 2 pour le no d'enregistrement
* 2 pour la somme ASCII
***********************************
       LI    R2,TEXT28
       LI    R3,ADD28
       BL    @AFFICH
       LI    R2,712
       LI    R7,709
       MOV   @TOT,R4
       BL    @DECI
* tri numerique sommes ascii
* methode shell-metzner
       MOV   @TOT,@ECART
TRIA1  CLR   @X
       MOV   @TOT,@K
* coupe en deux, sortie si=0
       MOV   @ECART,R15
       SRL   R15,1
       MOV   R15,@ECART
       CI    R15,1
       JL    INDENR
       INC   @X
       S     @ECART,@K
TRIA2  MOV   @X,@Y
TRIA3  MOV   @Y,@M
       A     @ECART,@M
* reperage enr. Y dans "stock"
       LI    R1,STOCK+2
       MOV   @Y,R12
       DEC   R12
       SLA   R12,2
       A     R12,R1
       MOV   *R1,@SOMME1
* reperage enr. M dans "stock"
       LI    R2,STOCK+2
       MOV   @M,R12
```

```
       DEC   R12
       SLA   R12,2
       A     R12,R2
       MOV   *R2,@SOMME2
* comparaison des deux enr.
       C     @SOMME1,@SOMME2
       JLE   TRIA4
* invertion des enregistrements
       DECT  R1
       MOV   *R1+,@INVER1
       MOV   *R1,@INVER2
       DECT  R1
       DECT  R2
       MOV   *R2+,*R1+
       MOV   *R2,*R1
       DECT  R2
       MOV   @INVER1,*R2+
       MOV   @INVER2,*R2
* suite du tri
       S     @ECART,@Y
       JH    TRIA3
TRIA4  INC   @X
       C     @X,@K
       JH    TRIA1
       JMP   TRIA2
* initialisation "index"
INDENR LI    R1,STOCK
       LI    R2,INDEX
       CLR   R3
IND1   MOV   *R1+,*R2+
       INC   R3
       C     R3,@TOT
       JH    IND2
       INCT  R1
       JMP   IND1
* ouverture fichier "index"
IND2   MOV   @CHTRI,@CHTRI
       JNE   IND4+4
       CLR   @LSAIS
       CLR   @NUMENR
       LI    R1,DSK2
       BL    @SATTR
       LI    R2,20
       BL    @OPEN
* mise en mode output
       LI    R0,PAB+1
       LI    R1,>0A00
       BLWP  @VSBW
* enregistrement de "index"
       CLR   R12
       LI    R1,INDEX
IND3   LI    R0,TPAB
       LI    R2,254
       BLWP  @VMBW
       A     R2,R12
       SRL   R2,1
       A     R2,@LSAIS
       BL    @PRINT
       BL    @DSR
       C     @LSAIS,@TOT
       JHE   IND4
* enregistrement suivant

       INC   @NUMENR
       LI    R0,PAB+6
```

```
       LI   R1,NUMENR
       LI   R2,2
       BLWP @VMBW
       LI   R1,INDEX
       A    R12,R1
       JMP  IND3
* operation terminee
IND4   BL   @OVER
       LI   R2,TEXT29
       LI   R3,ADD29
       BL   @AFFICH
       BL   @ENTER
       CLR  R12
       B    @FICH
****************************
* supprime l'appartenance
****************************
APPGR  LI   R0,568
       LI   R1,>3E00
```

```
       BLWP @VSBW
       LI   R2,TEXT46
       LI   R3,ADD46
       BL   @AFFICH
APP    BL   @ARRET
       CB   R4,@OUI
       JEQ  APP1
       CB   R4,@NON
       JNE  APP
       B    @NOMGR
APP1   LI   R2,TEXTOP
       LI   R3,ADD46+2
       BL   @AFFICH
* mode update
       LI   R0,PAB+1
       LI   R1,>0900
       BLWP @VSBW
       CLR  @NUMENR
LOOP1  INC  @NUMENR
```

```
       C    @NUMENR,@TOT
       JH   FINAPP
       BL   @NUM
       BL   @INPUT
       BL   @DSR
* charge inactif + 8 fois non
       LI   R0,TPAB+118
       LI   R1,GROFF
       LI   R2,9
       BLWP @VMBW
* reenregistrement
       BL   @NUM
       BL   @PRINT
       BL   @DSR
       JMP  LOOP1
* termine
FINAPP BL   @OVER
       BL   @CLEAR1
       B    @FINI2
```

Source "FICH3"

```
TEXT30 TEXT -'XXXXXXXXXXXXXXXXXXXXXXXXX '
       TEXT -'X IMPRESSION DU FICHIER X '
       TEXT -'XXXXXXXXXXXXXXXXXXXXXXXXX '
       TEXT -'OPTIONS D''IMPRESSION '
       TEXT -'~~~~~~~~~~~~~~~~~~~~ '
       TEXT -'1 Fichier organise '
       TEXT -'2 Etiquettes '
       TEXT -'3 Retour '
ADD30  DATA 47,87,127,369,409,490,530,570,0
ENTET1 TEXT ' No     NOM   PRENOM                  '
       TEXT ' ADRESSE'
       TEXT '                    CODE   VILLE      '
       TEXT '          '
       TEXT '       TEL              BOX       '
       TEXT 'COTISATION   '
       TEXT '          '
ENTET2 TEXT '---  --------------------------- '
       TEXT '---------'
       TEXT '------------------  ----- '
       TEXT '------------------'
       TEXT '--- --------------- --------   '
       TEXT '----------    '
NET    TEXT '                                     '
       TEXT '       '
TEXT31 TEXT '****** IMPRESSION EN COURS *****'
       TEXT -'* '
TEXT32 TEXT '   VOULEZ-VOUS UN TITRE ? O/N     '
       TEXT -'   '
TEXT33 TEXT '  AVEC DES GROS CARACTERES? O/N'
       TEXT -'     '
TEXT34 TEXT 'XXXXXX IMPRESSION TERMINEE XXXXX'
       TEXT -'X '
       TEXT -'<ENTER> '
ADD34  DATA 683,776,0
TEXT35 TEXT '+   centrage   +   centrage   +'
       TEXT -'     '
ADD35  DATA 683,0
TEXT36 TEXT '  VOULEZ-VOUS UN SOUS TITRE? O/N'
       TEXT -'     '
TEXT37 TEXT ' ALLUMEZ L''IMPRIMANTE ... <ENTER'
       TEXT -'>    '
```

```
TEXT38 TEXT 'VOUS SELECTIONNEZ DES GROUPES ? O/'
       TEXT -'N '
TEXT39 TEXT ' CHOISISSEZ O/N POUR CHAQUE GROUP'
       TEXT -'E   '
MODIMP TEXT -'   MODIFICATION DU NOM? O/N   '
SELECT TEXT ' IMPRESSION SELECTIVE PAR No? O/'
       TEXT -'N '
NUMERO TEXT '   No DE FICHE: 000 (ordre alpha'
       TEXT -') '
NOUVEL TEXT -'UN NOUVEAU CHAPITRE ? O/N '
ADDTER DATA 932,0
ININUM DATA >3030,>3000
* caracteres compresses
COMPR  DATA >1B43
* caracteres dilates et gras
BIGCAR DATA >1B0E,>1B23
* fin caracteres dilates,gras+mode normal
FINBIG DATA >1B0F,>1B24,>1B4E
TIMP   EQU >1200
DRAP10 DATA 0
INDX2  DATA 0
CHOIXG DATA >4E4E,>4E4E,>4E4E,>4E4E
       EVEN
TLIGNE BSS 136
IMP    DATA >0012,>1200,>0088,>0000
       BYTE >00
NOMIMP BYTE >00
       DATA >2020,>2020,>2020,>2020,>2020,>2020
       DATA >2020,>2020
       DATA >2020,>2020,>2020,>2020,>2020,>2020
       DATA >2020,>2020
* s/p lecture informations
READ   LI   R0,TPAB
       LI   R1,GROUPE
       LI   R2,94
       BLWP @VMBR
       LI   R0,TPAB+94
       LI   R1,NOMIMP
       LI   R2,33
       BLWP @VMBR
       RT
```

```
* s/p saisie info tpab
TRANS  BLWP @VSBR
       CB   R1,@POINT
       JEQ  RET4
       MOVB R1,*R3+
       INC  R0
       DEC  R2
       JEQ  RET4
       JMP  TRANS
RET4   RT
* s/p efface tligne
CLSTL  CLR  R2
       LI   R1,>2000
BLANC1 MOVB R1,@TLIGNE(R2)
       INC  R2
       CI   R2,136
       JNE  BLANC1
       RT
* verif. "index" deja charge
IMPRES SETO @DRAP10
       MOV  @INDX2,@INDX2
       JNE  IMPR1
       B    @GERER
* ouverture fichier "donnees"
IMPR1  LI   R1,DSK1
       BL   @SATTR
       LI   R2,22
       BL   @OPEN
* saisie enr.0
       CLR  @NUMENR
       BL   @INPUT
       BL   @DSR
       BL   @READ
* nom imprimante
       BL   @CLEAR1
       LI   R2,TEXT0
       LI   R3,ADD0
       BL   @AFFICH
       LI   R1,NOMIMP+1
       LI   R2,32
       LI   R0,524
       BLWP @VMBW
       LI   R2,MODIMP
       LI   R3,ADD14
       BL   @AFFICH
ARR16  BL   @ARRET
       CB   R4,@NON
       JEQ  AR0
* modif du nom
       CB   R4,@OUI
       JNE  ARR16
       CLR  @DRAP1
       CLR  @DRAP2
       SETO @DRAP4
       BL   @CLS24
       CLR  R12
       B    @IMPR+4
AR0    BL   @CLEAR
       CLR  @DRAP4
       LI   R2,TEXT30
       LI   R3,ADD30
       BL   @AFFICH
ARR7   BL   @ARRET
       CB   R4,@DEUX
       JNE  AR
       B    @ETIQ
AR     CB   R4,@UN
       JEQ  AR2
       CB   R4,@TROIS
       JNE  ARR7
       CLR  @DRAP10
       BL   @OVER
       B    @FICH
* efface option 2 et 3
AR2    CLR  R12
       CLR  R15
       LI   R0,530
       LI   R1,NET
       LI   R2,12
       BLWP @VMBW
       LI   R0,570
       LI   R1,NET
       LI   R2,8
       BLWP @VMBW
* impression selective
       LI   R2,SELECT
       LI   R3,ADD35
       BL   @AFFICH
STOP   BL   @ARRET
       CB   R4,@NON
       JEQ  OPGR
       CB   R4,@OUI
       JNE  STOP
       SETO @DRAP6
       JMP  AR8
* option groupes
OPGR   LI   R2,TEXT38
       LI   R3,ADD35
       BL   @AFFICH
       CLR  @DRAP6
ARR10  BL   @ARRET
       CB   R4,@NON
       JEQ  AR8
       CB   R4,@OUI
       JNE  ARR10
       SETO R15
CHGR   LI   R2,TEXT39
       LI   R3,ADD35
       BL   @AFFICH
       CLR  R8
       CLR  @LGR
AR7    LI   R0,770
       LI   R1,GROUPE
       A    @LGR,R1
       LI   R2,11
       BLWP @VMBW
       SETO @DRAP2
       SETO @DRAP4
       LI   R0,782
       LI   R2,1
       MOV  R2,@LSAIS
       LI   R7,CHOIXG
       A    R8,R7
       BL   @ACCEPT
       A    @D11,@LGR
       INC  R8
       C    R8,@NGR
       JL   AR7
AR8    LI   R0,760
       LI   R1,NET
       LI   R2,40
       BLWP @VMBW
       LI   R2,TEXT37
       LI   R3,ADD35
       BL   @AFFICH
       BL   @ENTER
       MOV  @DRAP11,@DRAP11
       JEQ  AR11
       B    @PLACE
AR1    B    @IMPR0
* ouvre l'imprimante
AR11   LI   R1,IMP
       BL   @SATTR
       LI   R2,42
       BL   @OPEN
AR12   LI   R2,TEXT32
       LI   R3,ADD35
       BL   @AFFICH
ARR8   BL   @ARRET
       CB   R4,@NON
       JEQ  AR1
       CB   R4,@OUI
       JNE  ARR8
       LI   R2,TEXT33
       LI   R3,ADD35
       BL   @AFFICH
       CLR  R14
ARR9   BL   @ARRET
       CB   R4,@NON
       JEQ  AR3
       CB   R4,@OUI
       JNE  ARR9
* caracteres doubles et gras
       LI   R0,TIMP
       LI   R1,BIGCAR
       LI   R2,4
       BLWP @VMBW
       LI   R0,PAB+5
       LI   R1,>0400
       BLWP @VSBW
       BL   @PRINT
       BL   @DSR
* saisie titre
AR3    BL   @OVER
       LI   R2,TEXT35
       LI   R3,ADD35
       BL   @AFFICH
       BL   @CLSTL
       LI   R2,40
       MOV  R2,@LSAIS
       LI   R0,760
       LI   R7,TLIGNE
       SETO @DRAP4
       CLR  @DRAP2
       BL   @ACCEPT
       DEC  R2
       MOV  R2,R13
* reouvre dis/var(R2)
       LI   R0,PAB+5
       MOV  R2,R1
       SWPB R1
       BLWP @VSBW
```

```
       BL    @DSR
* impression
       LI    R0,TIMP
       LI    R1,TLIGNE
       BLWP  @VMBW
       BL    @PRINT
       BL    @DSR
       MOV   R14,R14
       JNE   AR9
* soulignement
       LI    R0,TIMP
AR4    BLWP  @VSBR
       CB    R1,@BLANC
       JNE   AR5
       INC   R0
       JMP   AR4
AR5    LI    R1,>7E00
AR6    BLWP  @VSBW
       INC   R0
       DEC   R13
       JNE   AR6
       BL    @PRINT
       BL    @DSR
* lignes blanches
       BL    @CLSTI
       BL    @CLSTP
       LI    R1,1
       BL    @SAUTLI+4
       BL    @DSR
       CB    R4,@NON
       JEQ   IMPR0
* fin gros caracteres gras
       LI    R0,TIMP
       LI    R1,FINBIG
       LI    R2,4
       BLWP  @VMBW
       LI    R0,PAB+5
       LI    R1,>0400
       BLWP  @VSBW
       BL    @PRINT
       BL    @DSR
       BL    @OVER
* reouverture mode dis/var 136
       LI    R1,IMP
       LI    R2,42
       BL    @OPEN
* sous titre
IMPR0  LI    R0,760
       LI    R1,NET
       LI    R2,40
       BLWP  @VMBW
       LI    R2,TEXT36
       LI    R3,ADD35
       BL    @AFFICH
       BL    @ARRET
       CB    R4,@NON
       JEQ   AR9
       SETO  R14
       B     @AR3+4
AR9    LI    R1,NET
       LI    R0,680
       LI    R2,40
       BLWP  @VMBW
       LI    R2,TEXT31
       LI    R3,ADD35
       BL    @AFFICH
       LI    R0,760
       LI    R1,NET
       LI    R2,40
       BLWP  @VMBW
* caracteres compresses 136/ligne
       LI    R0,TIMP
       LI    R1,COMPR
       LI    R2,2
       BLWP  @VMBW
       LI    R0,PAB+5
       LI    R1,>0200
       BLWP  @VSBW
       BL    @PRINT
       BL    @DSR
* imprime l'en tete
       LI    R0,PAB+5
       LI    R1,>8800
       BLWP  @VSBW
       LI    R3,2
       LI    R0,TIMP
       LI    R1,ENTET1
       LI    R2,136
ENT2   BLWP  @VMBW
       BL    @DSR
       LI    R1,ENTET2
       DEC   R3
       JNE   ENT2
       BL    @INIDSK
       CLR   @LGR
* saisie par numero
       MOV   @DRAP6,@DRAP6
       JEQ   TRITEM
       SETO  R12
       LI    R2,NUMERO
       LI    R3,ADD35
       BL    @AFFICH
       LI    R2,TEXTG2
       LI    R3,ADDTER
       BL    @AFFICH
MOUV   LI    R2,ININUM
       LI    R3,TLIGNE
MOUV1  MOVB  *R2+,*R3+
       JNE   MOUV1
       CLR   @DRAP2
       SETO  @DRAP4
       LI    R0,699
       LI    R7,TLIGNE
       LI    R2,3
       MOV   R2,@LSAIS
       BL    @ACCEPT
       CI    R2,3
       JNE   MOUV
* transforme ascii en hexa
       LI    R2,-48
       SWPB  R2
       LI    R0,TLIGNE
       CLR   R1
       MOVB  *R0+,R1
       AB    R2,R1
       MOV   R1,@NUMENR
       MOVB  *R0+,R1
       AB    R2,R1
       SRL   R1,4
       A     R1,@NUMENR
       AB    R2,*R0
       CLR   R1
       MOVB  *R0,R1
       SWPB  R1
       A     R1,@NUMENR
       MOV   @NUMENR,@NUMENR
       JEQ   MOUV
       JMP   AUTRE+4
* si tri temporaire
TRITEM C     @TRI,@TOT
       JEQ   AUTRE
       MOV   @CHTRI,@CHTRI
       JEQ   AUTRE
       MOV   @TRI,@LSAIS
       MOV   @TOT,@TRI
* saisie enregistrements 1 a tot
AUTRE  INC   @NUMENR
       C     @NUMENR,@TOT
       JLE   IMPR2
       MOV   @CHTRI,@CHTRI
       JEQ   AUTR1
       MOV   @LSAIS,@TRI
AUTR1  B     @FINIMP
IMPR2  BL    @CLSTL
       BL    @PASS
IMPR3  BL    @INPUT
       BL    @DSR
* controle option groupe
       MOV   R15,R15
       JEQ   IMPR5
       LI    R0,TPAB+118
       LI    R1,BUFGR
       LI    R2,9
       BLWP  @VMBR
       CLR   @NLIG
       LI    R1,BUFGR+1
       LI    R2,CHOIXG
CONTR  CB    *R2,@OUI
       JEQ   CONTR1
       INC   R2
       INC   R1
       JMP   CONTR2
CONTR1 CB    *R2+,*R1+
       JEQ   IMPR5
CONTR2 INC   @NLIG
       C     @NLIG,@NGR
       JL    CONTR
       MOV   @DRAP11,@DRAP11
       JNE   IMPR6
       JMP   AUTRE
IMPR6  B     @LOOP
IMPR7  B     @PLACE1
* no dans tligne
IMPR5  MOV   @DRAP11,@DRAP11
       JNE   IMPR7
       INC   @LGR
       MOV   @LGR,R4
       LI    R2,TIMP+2
       LI    R7,TIMP-1
       BL    @DECI
       LI    R0,TIMP
       LI    R1,TLIGNE
```

```
       LI   R2,3
       BLWP @VMBR
* nom prenom
       LI   R0,TPAB+4
       LI   R3,TLIGNE+5
       LI   R2,28
       BL   @TRANS
* adresse
       LI   R0,TPAB+32
       LI   R3,TLIGNE+33
       LI   R2,28
       BL   @TRANS
* code postal
       LI   R0,TPAB+60
       LI   R3,TLIGNE+62
       LI   R2,5
       BL   @TRANS
* ville
       LI   R0,TPAB+65
       LI   R3,TLIGNE+68
       LI   R2,20
       BL   @TRANS
* tel
       LI   R0,TPAB+85
       LI   R3,TLIGNE+89
       LI   R2,15
       BL   @TRANS
* box
       LI   R0,TPAB+110
       LI   R3,TLIGNE+105
       LI   R2,8
       BL   @TRANS
* cotisation
       LI   R0,TPAB+100
       LI   R3,TLIGNE+115
       LI   R2,4
       BL   @TRANS
* no de licence
       LI   R0,TPAB+104
       LI   R3,TLIGNE+121
       LI   R2,6
       BL   @TRANS
* tligne dans timp
       LI   R0,TIMP
       LI   R1,TLIGNE
       LI   R2,136
       BLWP @VMBW
* init pab imprimante
       BL   @INIIMP
       BL   @SATTR
       BL   @PRINT
       BL   @DSR
*reinit pab fichier
       BL   @INIDSK
       BL   @SATTR
       MOV  R12,R12
       JNE  AUTR2
       B    @AUTRE
AUTR2  B    @MOUV
FINIMP BL   @OVER
       BL   @CLS24
       BL   @REMISE
       BL   @OVER
FINI2  LI   R2,TEXT34
       LI   R3,ADD34
       BL   @AFFICH
       BL   @ENTER
       CLR  R12
       CLR  @DRAP2
       CLR  @DRAP4
       CLR  @DRAP6
       CLR  @DRAP10
       B    @FICH
* nouveau chapitre?
NOUVE  LI   R2,NOUVEL
       LI   R3,ADDRET
       BL   @AFFICH
STOP1  BL   @ARRET
       CB   R4,@NON
       JEQ  FINIMP
       CB   R4,@OUI
       JNE  STOP2
       BL   @CLS24
       CLR  R12
       BL   @REMISE
       B    @AR12
* s/p mode normal imprimante
REMISE MOV  R11,@SAVRET
       BL   @INIIMP
       BL   @SATTR
       LI   R0,FINBIG+2
       LI   R1,NORM
       LI   R2,4
       BLWP @VMBW
       LI   R0,PAB+5
       LI   R1,>0400
       BLWP @VSBW
       BL   @PRINT
       BL   @DSR
       MOV  @SAVRET,R11
       RT
```

```
***************
* ETIQUETTES
***************
TEXT40 TEXT -'ETIQUETTES D''ADRESSES '
       TEXT -'~~~~~~~~~~~~~~~~~~~~~ '
       TEXT -'MISE EN PLACE '
       TEXT -'1-pas de modification '
       TEXT -'2-modification '
       TEXT -'3-retour '
ADD40  DATA 369,409,533,609,649,689,0
TEXT41 TEXT -'GROUPES A IMPRIMER '
       TEXT -'~~~~~~~~~~~~~~~~~~ '
ADD41  DATA 450,490,0
TEXT42 TEXT -'MODIFICATION DE MISE EN PLACE '
       TEXT -'  Marge de gauche :      '
       TEXT -'  Ecart central    : '
       TEXT -'      Sauts de lignes : '
       TEXT -'Nb d''etiq/ligne : '
ADD42  DATA 525,609,649,685,731,0
FORM1  TEXT 'TITRE..NOM...PRENOM.............'
FORM2  TEXT 'ADRESSE.........................'
FORM3  TEXT 'CODE.POSTAL....VILLE............'
TEXT43 TEXT -'LIGNE D''IMPRESSION TROP LONGUE  '
TEXT44 TEXT -'  EMPLACEMENT CORRECT? O/N    '
TEXT45 TEXT '  IMPRESSION CARATERES GRAS? O/'
       TEXT -'N  '
* code imprimante pour saut de ligne [CHR$(10)]
HEX10  DATA >0A0A,>0A0A,>0A0A,>0A0A,>0A0A
DRAP11 DATA 0
* options etiquettes*
MG     DATA 0
ENTRE  DATA 4
SAUT   DATA 5
NBETIQ DATA 2
*********************
TOTAL  DATA 72
EMPLAC DATA 0
* nb de colonnes de l'imprimante
MAXI   DATA 80
       EVEN
* s/p impression de la ligne
IMPLIG MOV  R11,@SAVRET
       LI   R0,TIMP
       LI   R1,TLIGNE
       MOV  @TOTAL,R2
       BLWP @VMBW
       BL   @PRINT
       BL   @DSR
       BL   @CLSTL
       MOV  @SAVRET,R11
       RT
* s/p init pab imprimante
INIIMP LI   R0,PAB
       LI   R1,IMP
       LI   R2,42
       BLWP @VMBW
       RT
* s/p init pab disquette
INIDSK LI   R0,PAB
       LI   R1,DSK1
       LI   R2,22
       BLWP @VMBW
       RT
* s/p mode dis/var (total)
DISVAR LI   R0,PAB+5
       MOV  @TOTAL,R1
       SWPB R1
       BLWP @VSBW
       RT
* s/p sauts de lignes
SAUTLI MOV  @SAUT,R1
       LI   R0,PAB+5
       MOV  R1,R2
       SWPB R1
       BLWP @VSBW
       LI   R0,TPAB
       LI   R1,HEX10
       BLWP @VMBW
       RT
* s/p de placement
PLACER LI   R0,TIMP
       A    @MG,R0
       BLWP @VMBW
       A    R2,R0
```

```
        A     @ENTRE,R0
        BLWP  @VMBW
        RT
* s/p efface timp
CLSTP   LI    R0,TIMP
        LI    R1,TLIGNE
        LI    R2,80
        BLWP  @VMBW
        RT
ETIQ    BL    @CLEAR1
        LI    R2,TEXT40
        LI    R3,ADD40
        BL    @AFFICH
ARR13   BL    @ARRET
        CB    R4,@UN
        JEQ   ETIQ2
        CB    R4,@DEUX
        JEQ   ETIQ1
        CB    R4,@TROIS
        JNE   ARR13
        B     @AR0
* modif de mise en place
ETIQ1   LI    R2,TEXT42
        LI    R3,ADD42
        BL    @AFFICH
        SETO  @DRAP2
        SETO  @DRAP4
        SETO  @DRAP6
MG1     CLR   @NLIG
        CLR   @LGR
        LI    R6,48
        A     R6,@MG
        A     R6,@ENTRE
        A     R6,@SAUT
        A     R6,@NBETIQ
        LI    R5,4
        LI    R2,1
        MOV   R2,@LSAIS
MG2     LI    R0,629
        A     @NLIG,R0
        LI    R1,MG+1
        A     @LGR,R1
        MOV   R1,R7
        BL    @ACCEPT
        A     @D40,@NLIG
        INCT  @LGR
        DEC   R5
        JNE   MG2
* tranforme en nombre
        LI    R6,-48
        A     R6,@MG
        A     R6,@ENTRE
        A     R6,@SAUT
        A     R6,@NBETIQ
        MOV   @D33,@TOTAL
        MOV   @NBETIQ,R2
M0      DEC   R2
        JEQ   M1
        A     @D33,@TOTAL
        JMP   M0
M1      A     @MG,@TOTAL
        A     @ENTRE,@TOTAL
        C     @TOTAL,@MAXI
        JLE   ETIQ2
* impossible
        LI    R2,TEXT43
        LI    R3,ADD14
        BL    @AFFICH
        BL    @ARRET
        BL    @CLS24
        JMP   MG1
* choix des groupes
ETIQ2   BL    @CLEAR1
        LI    R2,TEXT41
        LI    R3,ADD41
        BL    @AFFICH
        SETO  @DRAP11
        CLR   @DRAP6
        B     @CHGR
* ouvrir l'imprimante
PLACE   LI    R1,IMP
        LI    R2,43
        BL    @OPEN
        LI    R2,TEXT45
        LI    R3,ADD35
        BL    @AFFICH
ARR14   BL    @ARRET
        CB    R4,@NON
        JEQ   MG3
        CB    R4,@OUI
        JNE   ARR14
* init caracteres gras
        LI    R0,TIMP
        LI    R1,BIGCAR+2
        LI    R2,2
        BLWP  @VMBW
        LI    R0,PAB+5
        LI    R1,>0200
        BLWP  @VSBW
        BL    @PRINT
        BL    @DSR
MG3     BL    @DISVAR
* etiquette test
        BL    @CLSTL
        BL    @CLSTP
        LI    R1,FORM1
        LI    R2,33
        BL    @PLACER
        LI    R2,TEXT31
        LI    R3,ADD35
        BL    @AFFICH
* 1ere ligne
        BL    @PRINT
        BL    @DSR
* 2ieme ligne
        BL    @CLSTP
        LI    R1,FORM2
        LI    R2,33
        BL    @PLACER
        BL    @DSR
* 3ieme ligne
        BL    @CLSTP
        LI    R1,FORM3
        LI    R2,33
        BL    @PLACER
        BL    @DSR
* sauts
        MOV   @SAUT,@SAUT
        JEQ   CORR
        BL    @SAUTLI
        BL    @DSR
        BL    @DISVAR
CORR    LI    R2,TEXT44
        LI    R3,ADD14
        BL    @AFFICH
ARR15   BL    @ARRET
        CB    R4,@OUI
        JEQ   MG4
        CB    R4,@NON
        JNE   ARR15
        BL    @CLEAR1
        B     @ETIQ1
MG4     CLR   @DRAP2
        CLR   @DRAP4
        CLR   @DRAP6
        CLR   @NUMENR
        CLR   @LSAIS
        MOV   @NBETIQ,R13
        BL    @CLS24
        BL    @CLSTL
        LI    R2,TEXT31
        LI    R3,ADD35
        BL    @AFFICH
        CLR   R8
* enregistrements
LOOP    INC   @NUMENR
        C     @NUMENR,@TOT
        JH    L1
        BL    @INIDSK
        BL    @SATTR
        SETO  R15
        BL    @NUM
        B     @IMPR3
L1      CI    R13,1
        JEQ   PLACE0
        B     @FIN
* stockage
PLACE0  SETO  R8
        BL    @CLSTL
        LI    R0,TPAB
        LI    R1,TLIGNE
        LI    R2,127
        BLWP  @VMBW
PLACE1  LI    R1,STOCK
        A     @LSAIS,R1
        LI    R2,118
        LI    R0,TPAB
        BLWP  @VMBR
        A     R2,@LSAIS
        DEC   R13
        JNE   LOOP
        BL    @INIIMP
        BL    @SATTR
        BL    @DISVAR
* titre
PLACE2  LI    R3,STOCK
        MOV   R3,@EMPLAC
        LI    R4,TLIGNE
        A     @MG,R4
        MOV   @NBETIQ,R5
PLACE3  LI    R2,4
PL1     CB    *R3,@POINT
```

```
       JEQ  PLACE4
       MOVB *R3+,*R4+
       DEC  R2
       JNE  PL1
* nom prenom
PLACE4 LI   R2,28
       INC  R4
       MOV  @EMPLAC,R3
       AI   R3,4
PL2    CB   *R3,@POINT
       JEQ  PL3
       MOVB *R3+,*R4+
       DEC  R2
       JNE  PL2
PL3    DEC  R5
       JEQ  PL4
       LI   R4,TLIGNE+33
       A    @ENTRE,R4
       LI   R3,STOCK+118
       MOV  R3,@EMPLAC
       JMP  PLACE3
PL4    BL   @IMPLIG
* adresse
       LI   R3,STOCK+32
       LI   R4,TLIGNE
       A    @MG,R4
       MOV  @NBETIQ,R5
PLACE5 LI   R2,28
PL5    CB   *R3,@POINT
       JEQ  PL6
       MOVB *R3+,*R4+
       DEC  R2
       JNE  PL5
PL6    DEC  R5
       JEQ  PL7
       LI   R3,STOCK+150
       LI   R4,TLIGNE+33
       A    @ENTRE,R4
       JMP  PLACE5
PL7    BL   @IMPLIG
* code et ville
       LI   R3,STOCK+60
       MOV  R3,@EMPLAC
       LI   R4,TLIGNE
       A    @MG,R4
       MOV  @NBETIQ,R5
PLACE6 LI   R2,5
PL8    CB   *R3,@POINT
       JEQ  PLACE7
       MOVB *R3+,*R4+
       DEC  R2
       JNE  PL8
PLACE7 LI   R2,20
       INC  R4
       MOV  @EMPLAC,R3
       AI   R3,5
PL9    CB   *R3,@POINT
       JEQ  PL10
       MOVB *R3+,*R4+
       DEC  R2
       JNE  PL9
PL10   DEC  R5
       JEQ  PL11
       LI   R4,TLIGNE+33
       A    @ENTRE,R4
       LI   R3,STOCK+178
       MOV  R3,@EMPLAC
       JMP  PLACE5
PL11   BL   @IMPLIG
* sauts
       MOV  @SAUT,@SAUT
       JEQ  SUIVRE
       BL   @SAUTLI
       BL   @DSR
       BL   @DISVAR
SUIVRE CLR  @LSAIS
       MOV  @NBETIQ,R13
       MOV  R8,R8
       JNE  FIN
       B    @LOOP
FIN    BL   @INIDSK
       BL   @SATTR
       CLR  @DRAP11
       B    @FINIMP
```

Source "FICH4"

```
********************
* MODIFIER UN NOM
********************
* s/p d'attente
DELAI  CLR  R15
       INC  R15
       JNE  DELAI+2
       RT
DRAP4  DATA 0
TEXT11 TEXT -'   modifier une fiche    '
       TEXT -'Rentrez le nom de la personne '
       TEXT -'que vous voulez modifier. '
       TEXT -'NOM Prenom: '
ADD11  DATA 48,325,367,520,0
TEXT12 TEXT 'NOM INCONNU AU FICHIER!...<ENTER'
       TEXT -'> '
TEXT13 TEXT -'LE FICHIER EST VIDE!...<ENTER> '
TEXT14 TEXT -'NOM DEJA SUR FICHIER!...<ENTER> '
ADD14  DATA 925,0
TEXT15 TEXT -'UNE AUTRE MODIFICATION ? O/N '
TEXTI  TEXT -'inactif '
ADDI   DATA 229,0
TEXTA  TEXT -' actif '
TEXTRI TEXT 'SI VOUS AVEZ MODIFIE LE NOM OU LE'
       TEXT -' PRENOM '
       TEXT 'D''UNE PERSONNE, FAITES FONCTIONNE'
       TEXT -'R LE '
       TEXT -'PROGRAMME DE TRI IMPERATIVEMENT. '
       TEXT -'<ENTER> '
ADDTRI DATA 520,601,684,775,0
TEXTEX TEXT 'Dans  MODIFIER,EFFACER,AFFICHER, '
       TEXT -'vous '
       TEXT 'pouvez, pour la saisie, ne mettre '
       TEXT -'que '
       TEXT -'-les 3 premieres lettres du nom '
       TEXT -'-la premiere lettre du prenom '
       TEXT 'sans oublier l''espace entre '
       TEXT -'nom-prenom '
       TEXT -'<ENTER> '
ADDEX  DATA 441,481,521,561,601,775,0
       EVEN
******************
MODIF  BL   @CLEAR1
       LI   R2,TEXTEX
       LI   R3,ADDEX
       BL   @AFFICH
       BL   @ENTER
MODIF0 BL   @CLEAR1
       LI   R2,TEXT11
       LI   R3,ADD11
       BL   @AFFICH
* saisie du nom
MODIF1 LI   R1,>2E00
       BL   @INITAM
       LI   R0,532
       LI   R1,TAMPON+4
       MOV  R1,R7
       LI   R2,28
       BLWP @VMBW
       MOV  R2,@LSAIS
       CLR  @DRAP2
       BL   @ACCEPT
```

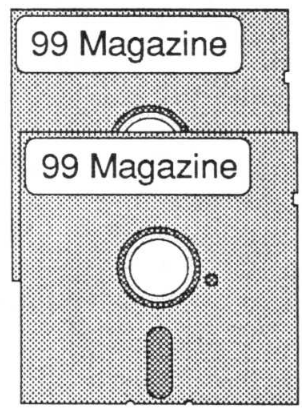
```
* transfert dans tpab
       LI    R0,TPAB+4
       LI    R1,TAMPON+4
       LI    R2,28
       BLWP  @VMBW
* somme codes ascii
       BL    @ADDIT
       MOV   @ASCII,@SOMME1
       CLR   @DRAP3
* recherche dans fichier
       BL    @RBLOC
       MOV   @DRAP,@DRAP
       JNE   AUTMOD
* transfert dans "tampon"
       LI    R0,TPAB
       LI    R1,TAMPON
       LI    R2,127
       BLWP  @VMBR
* verifie si effacement
       MOV   @DRAP7,@DRAP7
       JNE   MOD4
       MOVB  @BUFGR,R3
       CB    R3,@E
       JNE   MODIF2
       BL    @CLEAR1
       LI    R2,TEXT18
       LI    R3,ADD18
       BL    @AFFICH
ARR3   BL    @ARRET
       CB    R4,@NON
       JEQ   AUTMOD
       CB    R4,@OUI
       JNE   ARR3
* reintroduction
       LI    R0,TPAB+118
       MOVB  @A,R1
       BLWP  @VSBW
       SETO  @DRAP4
       B     @AJENR4
* affichage
MODIF2 MOV   @DRAP5,@DRAP5
       JNE   MOD3
       BL    @CLEAR1
       LI    R2,TEXT6
       LI    R3,ADD6
       BL    @AFFICH
* actif ou inactif
       MOVB  @BUFGR,R4
       CB    R4,@I
       JNE   ACTIF
       LI    R2,TEXTI
       LI    R3,ADDI
       BL    @AFFICH
       JMP   NUM1
ACTIF  LI    R2,TEXTA
       LI    R3,ADDI
       BL    @AFFICH
NUM1   MOV   @ALPHA,R4
       SETO  @DRAP4
       CLR   @DRAP3
       B     @AJOUT4
* autre modification ?
AUTMOD BL    @CLS24
       MOV   @DRAP5,@DRAP5
       JNE   MOD1
       MOV   @DRAP7,@DRAP7
       JNE   MOD5
       LI    R2,TEXT15
       LI    R3,ADD14
       BL    @AFFICH
       CLR   @DRAP
       CLR   @DRAP2
       BL    @DELAI
ARR1   BL    @ARRET
       CB    R4,@OUI
       JEQ   MOD2
       CB    R4,@NON
       JNE   ARR1
       BL    @CLEAR1
       LI    R2,TEXTRI
       LI    R3,ADDTRI
       BL    @AFFICH
       BL    @ENTER
       B     @OPT
MOD1   B     @AUTEFF
MOD2   B     @MODIF0
MOD3   B     @EFFA1
MOD4   MOV   @ALPHA,@NUMENR
       B     @RECH2
MOD5   B     @RECH6
******************
* EFFACER UN NOM
******************
DRAP5  DATA 0
TEXT16 TEXT -'   effacer une fiche    '
       TEXT -'Rentrez le nom de la personne '
       TEXT -'que vous voulez effacer. '
       TEXT -'NOM Prenom: '
ADD16  DATA 48,365,407,520,0
TEXT17 TEXT -'  UN AUTRE EFFACEMENT ?  O/N '
TEXT18 TEXT -'LA FICHE EST MARQUEE EFFACEMENT '
       TEXT -'REINTRODUCTION ? O/N '
ADD18  DATA 525,610,0
       EVEN
******************
EFFACE BL    @CLEAR1
       LI    R2,TEXTEX
       LI    R3,ADDEX
       BL    @AFFICH
       BL    @ENTER
EFFA2  BL    @CLEAR1
       LI    R2,TEXT16
       LI    R3,ADD16
       BL    @AFFICH
* saisie du nom
       SETO  @DRAP5
       B     @MODIF1
* autre effacement?
AUTEFF LI    R2,TEXT17
       LI    R3,ADD14
       BL    @AFFICH
       CLR   @DRAP
       CLR   @DRAP2
       CLR   @DRAP4
ARR2   BL    @ARRET
       CB    R4,@OUI
       JEQ   EFFA2
       CB    R4,@NON
       JNE   ARR2
       B     @OPT
* routine marque "E"
EFFA1  LI    R0,TPAB+118
       MOVB  @E,R1
       BLWP  @VSBW
```

```
* reenregistrement de la fiche
       SETO  @DRAP4
       B     @AJENR4
**********************
* MODIFIER UN GROUPE
**********************
E8375  BYTE 0
DRAP6  DATA 0
NOGR   DATA 0
TEXRET TEXT -'<BACK> = retour '
ADDRET DATA -926,0
TEXTEF TEXT -' effacer dernier groupe '
       TEXT -'Appuyez sur la  barre  <ESPACE> '
       TEXT -'pour effacer le dernier groupe. '
       TEXT -'<ENTER> = ARRET '
TEXTCO TEXT -'   corriger un groupe   '
       TEXT -'Rentrez le numero du groupe '
       TEXT -'que vous voulez corriger '
       TEXT -'No du groupe '
TEXTAJ TEXT -'   ajouter un groupe    '
       TEXT -'Rentrez le nom du nouveau groupe '
TEXNOM TEXT -'Nom du groupe '
```

```
       TEXT -'~~~~~~~~~~~~~~ '
ADDEF  DATA 48,285,325,663,0
ADDAJ  DATA 48,282,663,703,0
ADDCO  DATA 48,285,327,543,0
ADDNOM DATA 663,703,0
ADD13  DATA 403,443,483,523,563,603,643,683
       DATA 723,763,803,843,0
TEXT19 TEXT -'PLUS DE GROUPE LIBRE ... <ENTER> '
       EVEN
******************
MODGR  BL   @CLEAR1
       LI   R2,TEXTG1
       LI   R3,ADD13
       BL   @AFFICH
       LI   R2,TEXRET
       LI   R3,ADDRET
       BL   @AFFICH
INS1   LI   R3,8
       CLR  @LGR
       LI   R2,11
       LI   R0,487
* inscrire les groupes
INSGR  LI   R1,GROUPE
       A    @LGR,R1
       BLWP @VMBW
       A    @D40,R0
       A    @D11,@LGR
       DEC  R3
       JNE  INSGR
       MOV  @DRAP7,@DRAP7
       JNE  SAISNO
       MOV  @DRAP6,@DRAP6
       JNE  INS5
       SETO @DRAP6
* orientation
       CB   @RTCLA,@DEUX
       JEQ  CORNOM
       CB   @RTCLA,@UN
       JEQ  AJNOM
* effacer groupe
       LI   R2,TEXTEF
       LI   R3,ADDEF
       BL   @AFFICH
STOP3  BL   @ARRET
       BL   @BLOQ
       JNE  STOP3
       CB   R4,@BLANC
       JEQ  EFFGR
       CB   R4,@ENT
       JEQ  INS6
       CB   R4,@BACK
       JNE  STOP3
       CLR  @DRAP6
       B    @NOMGR
EFFGR  DEC  @NGR
       MOV  @NGR,R2
       MOVB @TROIS,@RTCLA
       MPY  @D11,R2
       LI   R1,GROUPE
       A    R3,R1
       MOV  R1,R5
       LI   R2,11
       LI   R3,NET
EFFG   MOVB *R3,*R1+
       DEC  R2
       JNE  EFFG
       CLR  @DRAP6
       JMP  INS1
INS6   CLR  @DRAP6
       B    @ENREG0
* ajouter un nom de groupe
AJNOM  MOVB @UN,@E8375
       LI   R2,TEXTAJ
       LI   R3,ADDAJ
       BL   @AFFICH
       JMP  SGROUP
INS5   B    @INS2
* corriger un nom de groupe
CORNOM LI   R2,TEXTCO
       LI   R3,ADDCO
       BL   @AFFICH
       MOVB @DEUX,@E8375
* saisie du No de groupe
SAISNO SETO @DRAP4
       SETO @DRAP2
       CLR  @NOGR
       LI   R2,1
       LI   R0,557
       LI   R7,NOGR+1
       MOV  R2,@LSAIS
       BL   @ACCEPT
       CB   @NOGR+1,@UN
       JL   SAISNO
       LI   R1,-48
       A    R1,@NOGR
       C    @NOGR,@NGR
       JH   SAISNO
       MOV  @DRAP7,@DRAP7
       JNE  INS4
* affichage et saisie groupe
       LI   R2,TEXNOM
       LI   R3,ADDNOM
       BL   @AFFICH
       MOV  @NOGR,R1
       MPY  @D11,R1     -> result. R2
       LI   R0,784
       LI   R1,GROUPE-11
       A    R2,R1
       LI   R2,11
       BLWP @VMBW
       MOV  @D11,@LSAIS
       CLR  @DRAP2
       MOV  R1,R7
       BL   @ACCEPT
       B    @INS1
INS4   B    @AFFIG1
* saisie du nouveau nom
SGROUP LI   R2,8
       C    @NGR,R2
       JHE  TOUSUT
       MOV  @D11,@LSAIS
       LI   R3,GROUPE
       MOV  @NGR,R1
       MPY  @D11,R1
       A    R2,R3
       MOV  R3,R7
       LI   R0,784
       CLR  @DRAP2
       SETO @DRAP4
       BL   @ACCEPT
       LI   R0,784
       BLWP @VSBR
       CB   R1,@BLANC
       JEQ  SGROUP
       INC  @NGR
       CLR  @DRAP6
       B    @INS1
* autre modification?
INS2   LI   R2,TEXT15
       LI   R3,ADD14
       BL   @AFFICH
ARR4   BL   @ARRET
       CB   R4,@NON
       JNE  INS3
       CLR  @DRAP6
       B    @ENREG0
INS3   CB   R4,@OUI
       JNE  ARR4
       CLR  @DRAP6
       MOVB @E8375,@RTCLA
       B    @MODGR
* tous les groupes sont utilises
TOUSUT LI   R2,TEXT19
       LI   R3,ADD14
       BL   @AFFICH
       BL   @ARRET
       CLR  @DRAP6
       B    @ENREG0
**************************
* PROGRAMME D'AFFICHAGE
**************************
DRAP7  DATA 0
DRAP8  DATA 0
DRAP9  DATA 0
TEXT20 TEXT -'AFFICHER UN NOM '
       TEXT -'~~~~~~~~~~~~~~~ '
       TEXT -'Rentrez le nom de la personne '
       TEXT -'que vous voulez afficher. '
       TEXT -'NOM Prenom : '
ADD20  DATA 251,291,365,407,520,0
TEXT21 TEXT -'AFFICHER UN GROUPE '
       TEXT -'~~~~~~~~~~~~~~~~~~ '
       TEXT -'No du groupe '
       TEXT -'a afficher : '
```

```
ADD21  DATA 248,288,503,543,0
TEXT22 TEXT -' effacement '
TEXT23 TEXT -'Enregistrement No     Marque: '
       TEXT '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'
       TEXT -'~~~~~~~~~~~~ '
       TEXT -'xxxxxxxxx
       TEXT -'x ordre x '
       TEXT -'x alpha x '
       TEXT -'x  000  x '
       TEXT -'xxxxxxxxx '
       TEXT -'Dans groupe '
       TEXT -'~~~~~~~~~~~~ '
ADD23  DATA
200,240,311,351,391,431,471,509,549,0
AFFECR DATA 322,402,482,562,568,642,722,727,802
TEXT24 TEXT -'  UN AUTRE AFFICHAGE ? O/N '
TEXT25 TEXT '<ENTER> autre fiche  <BACK> arre'
       TEXT -'t '
TEXT26 TEXT -'FIN DU FICHIER ... <ENTER> '
ADD26  DATA 527,0
       EVEN
```

```
*************
* une fiche
*************
RECH1  BL   @CLEAR1
       LI   R2,TEXTEX
       LI   R3,ADDEX
       BL   @AFFICH
       BL   @ENTER
RECH0  BL   @CLEAR1
       LI   R2,TEXT20
       LI   R3,ADD20
       BL   @AFFICH
       SETO @DRAP7
       B    @MODIF1
* No,actif,inactif,effacement
RECH2  BL   @CLEAR1
       LI   R2,TEXT23
       LI   R3,ADD23
       BL   @AFFICH
       C    @NUMENR,@TRI
       JH   RECH7
       MOV  @SECT,R4
       JMP  RECH8
RECH7  MOV  @NUMENR,R4
RECH8  LI   R2,220
       LI   R7,217
       BL   @DECI
       MOV  @NUMENR,R4
       LI   R2,436
       LI   R7,433
       BL   @DECI
       MOVB @BUFGR,R4
       CB   R4,@E
       JNE  ACT
       LI   R2,TEXT22
       LI   R3,ADDI
       BL   @AFFICH
       JMP  RECH3
ACT    CB   R4,@A
       JNE  INAC
       LI   R2,TEXTA
       LI   R3,ADDI
       BL   @AFFICH
       JMP  RECH3
INAC   LI   R2,TEXTI
       LI   R3,ADDI
       BL   @AFFICH
* affichage informations
RECH3  LI   R3,NBSAIS
       LI   R4,AFFECR
       LI   R5,TAMPON
RECH4  MOV  *R3+,R2
       JEQ  RECH5
       MOV  R5,R7
       MOV  *R4+,R0
       CLR  R6
RE4    MOVB *R7+,R1
       CB   R1,@POINT
       JEQ  RE5
       BLWP @VSBW
       INC  R0
       INC  R6
       C    R6,R2
       JL   RE4
RE5    A    R2,R5
       JMP  RECH4
* affichage des groupes
RECH5  LI   R3,BUFGR
       LI   R2,11
       CLR  @NLIG
       CLR  @LGR
       MOV  @NGR,R4
       INC  R4
RE1    DEC  R4
       JEQ  RECH6
       INC  R3
       LI   R0,589
       LI   R1,GROUPE-11
       A    @D11,@LGR
       CB   *R3,@OUI
       JNE  RE1
       A    @NLIG,R0
       A    @LGR,R1
       BLWP @VMBW
       A    @D40,@NLIG
       JMP  RE1
* autre fiche?
RECH6  MOV  @DRAP8,@DRAP8
       JNE  SUITFI
       LI   R2,TEXT24
       LI   R3,ADDRET
       BL   @AFFICH
ARR5   BL   @ARRET
       CB   R4,@NON
       JNE  RE2
       B    @OPT
RE2    CB   R4,@OUI
       JNE  ARR5
       CLR  @DRAP
       CLR  @DRAP2
       B    @RECH0
*******************
* tout le fichier
*******************
TOUTFI SETO @DRAP7
       SETO @DRAP8
       CLR  @NUMENR
* cas de tri temporaire
       C    @TRI,@TOT
       JEQ  TOUT1
       MOV  @CHTRI,@CHTRI
       JEQ  TOUT1
       MOV  @TRI,@LSAIS
       MOV  @TOT,@TRI
* saisie de la fiche
TOUT1  INC  @NUMENR
       C    @NUMENR,@TOT
       JH   FITERM
       BL   @PASS
       BL   @INPUT
       BL   @DSR
       LI   R0,TPAB
       LI   R1,TAMPON
       LI   R2,127
       BLWP @VMBR
* routine de recherche de groupe
       MOV  @DRAP9,@DRAP9
       JEQ  TOUT2
       MOV  @NOGR,R2
       CB   @BUFGR(R2),@OUI
       JNE  TOUT1
TOUT2  CLR  @DRAP2
       B    @RECH2
* <enter> ou <back>
SUITFI LI   R2,TEXT25
       LI   R3,ADD14
       BL   @AFFICH
       BL   @ENTER
       MOV  @DRAP2,@DRAP2
       JNE  TOUT1
       MOV  @CHTRI,@CHTRI
       JEQ  FITER1
       MOV  @LSAIS,@TRI
       B    @OPT
* fin du fichier
FITERM BL   @CLEAR1
       LI   R2,TEXT26
       LI   R3,ADD26
       BL   @AFFICH
       BL   @ARRET
       MOV  @CHTRI,@CHTRI
       JEQ  FITER1
       MOV  @LSAIS,@TRI
FITER1 B @OPT
*************
* un groupe
*************
AFFIGR BL   @CLEAR1
       LI   R2,TEXT21
       LI   R3,ADD21
       BL   @AFFICH
       SETO @DRAP6
       SETO @DRAP7
       B    @MODGR+4
* recherche des fiches
AFFIG1 SETO @DRAP9
       B    @TOUTFI
```




```
* s/p passage eventuel "index"
PASS   C    @NUMENR,@TRI
       JH   TOUT3
       LI   R1,INDEX
       MOV  @NUMENR,R2
       DEC  R2
       SLA  R2,1
       A    R2,R1
       MOV  *R1,@SECT
       JMP  TOUT4
TOUT3  LI   R1,NUMENR
TOUT4  LI   R0,PAB+6
       LI   R2,2
       BLWP @VMBW
       RT
```

Ticrok

Georges Goument

Je suis Ticrok le rouge. Si vous arrivez à me capturer, étant impulsif et coléreux de nature, je me détruirais en vous laissant le gain de la partie.

Ne rêvez pas trop quand même ; dans ma petite tête de monstre, j'ai plus d'un piège en réserve. Soyez assuré que, le moment venu, je saurais les utiliser contre vous.

Monstre je suis, mais bête... sûrement pas ! Bonne chance.

Comment me capturer ?

Après chacun de vos déplacements, vous laissez une barrière sur place. Les barrières, je ne peux les franchir, tout comme vous d'ailleurs, alors attention ; ne vous piégez pas vous-même.

Vous devrez me bloquer étroitement pour que j'explose de rage.

De temps en temps, je suis un peu farceur et, pour vous narguer, je m'isole moi-même. Je peux me promener tranquillement, hors d'atteinte. Dans ce cas, vous perdrez également la partie.

A vous de trouver la parade pour éviter qu'une telle situation se reproduise trop souvent.

Je suis quand même un chouette monstre pour vous révéler un de mes trucs. Malheureusement (pour vous), il y en a d'autres.

Vos déplacements s'effectuent à l'aide des touches 1 à 8. Le dessin situé sous le "terrain de jeu" vous indique les directions correspondantes.

Si vous me laissez m'isoler et que ma capture ne soit plus possible, appuyez sur "A".

Toutes les dix parties, le jeu s'arrête et affiche le résultat final.

99

Programme Ticrok

Basic TI

```
100 REM *************
110 REM *           *
120 REM *   TICROK  *
130 REM *           *
140 REM *************
150 REM *           *
160 REM *  basic TI *
170 REM *           *
180 REM *************
190 REM *           *
200 REM * copyright *
210 REM *           *
220 REM *99  Magazine*
230 REM *           *
240 REM * G. Goument *
250 REM *           *
260 REM *************
270 PP=0
280 P1=0
290 P2=0
300 MY=1
310 MP=1
320 Z=0
330 CALL CLEAR
340 RANDOMIZE
350 CALL SCREEN(8)
360 FOR I=1 TO 13
370 CALL COLOR(I,1,1)
380 NEXT I
390 FOR I=40 TO 48
400 READ A$
410 CALL CHAR(I,A$)
420 NEXT I
430 DATA 991818FFFF181899,0060601818060601,0018181818181899,000606181860608,800000FEFE00008
440 DATA 80606018180606,99181818181818,0106061818606,0100007F7F000001
450 CALL CHAR(33,"1818103854102844")
460 CALL CHAR(34,"3C7EDBDB7E18A542")
470 CALL COLOR(1,13,1)
480 CALL CHAR(58,"0")
490 CALL CHAR(64,"FFFFFFFFFFFFFFFF")
500 FOR I=72 TO 79
510 READ A$
520 CALL CHAR(I,A$)
530 NEXT I
540 DATA 1818103854102844,030309225609102,0010D021320C1723,020218162068D0C8,181810315F,C4C8305088422
550 DATA 10098A5E20E0D8,C0C03C64A7
560 CALL CHAR(80,"3C7EDBDB7E18A542")
570 CALL CHAR(88,"00000000AAFEAAAA")
580 CALL CHAR(120,"900250008014400A")
590 PRINT TAB(12);"8:1:2":TAB(12);":)*+:":TAB(12);"70(,3":TAB(12);":/.-:":TAB(12);"6:5:4":::
600 CALL HCHAR(1,3,72)
610 CALL HCHAR(1,30,80)
620 CALL HCHAR(1,6,58,21)
630 CALL HCHAR(17,6,58,8)
640 CALL HCHAR(17,19,58,8)
650 CALL VCHAR(2,6,58,15)
660 CALL VCHAR(2,26,58,15)
670 FOR I=7 TO 25
680 CALL VCHAR(2,I,64,15)
690 NEXT I
700 FOR I=9 TO 11
710 CALL COLOR(I,6,1)
720 NEXT I
730 V=0
740 W=0
750 V=INT((11-2+1)*RND)+2
760 W=INT((20-12+1)*RND)+12
770 CALL HCHAR(V,W,72)
780 VA=V
```

```
790 WA=W
800 Y=0
810 X=0
820 Y=INT((11-2+1)*RND
    )+2
830 X=INT((20-12+1)*RN
    D)+12
840 CALL GCHAR(Y,X,A)
850 IF A<>64 THEN 820
860 YA=Y
870 XA=X
880 CALL HCHAR(Y,X,80)
890 CALL COLOR(5,15,15
    )
900 CALL COLOR(6,14,15
    )
910 CALL COLOR(7,9,15)
920 CALL COLOR(8,7,15)
930 CALL COLOR(12,5,15
    )
940 CALL COLOR(2,16,13
    )
950 CALL COLOR(3,16,13
    )
960 CALL COLOR(4,16,13
    )
970 PP=PP+1
980 TJ=0
990 N=0
1000 OB=0
1010 ND=0
1020 O=0
1030 DO=INT((10-2+1)*R
     ND)+2
1040 O=O+1
1050 IF O=DO*2 THEN 10
     20
1060 H=0
1070 B=0
1080 G=0
1090 D=0
1100 IF O<DO THEN 1150
1110 H=INT((9-3+1)*RND
     )+3
1120 B=INT((17-11+1)*R
     ND)+11
1130 G=INT((13-7+1)*RN
     D)+7
1140 D=INT((25-19+1)*R
     ND)+19
1150 XA=X
1160 YA=Y
1170 VA=V
1180 WA=W
1190 V=0
1200 W=0
1210 Y=0
1220 X=0
1230 CALL KEY(3,T,S)
1240 IF S=0 THEN 1230
1250 IF T<>49 THEN 130
     0
1260 IF VA=2 THEN 1780
1270 V=VA-1
1280 W=WA
1290 GOTO 1720
1300 IF T<>50 THEN 136
     0
1310 IF VA=2 THEN 1780
1320 IF WA=25 THEN 178
     0
1330 V=VA-1
1340 W=WA+1
1350 GOTO 1720
1360 IF T<>51 THEN 141
     0
1370 IF WA=25 THEN 178
     0
1380 W=WA+1
1390 V=VA
1400 GOTO 1720
1410 IF T<>52 THEN 147
     0
1420 IF WA=25 THEN 178
     0
1430 IF VA=16 THEN 178
     0
1440 V=VA+1
1450 W=WA+1
1460 GOTO 1720
1470 IF T<>53 THEN 152
     0
1480 IF VA=16 THEN 178
     0
1490 V=VA+1
1500 W=WA
1510 GOTO 1720
1520 IF T<>54 THEN 158
     0
1530 IF VA=16 THEN 178
     0
1540 IF WA=7 THEN 1780
1550 V=VA+1
1560 W=WA-1
1570 GOTO 1720
1580 IF T<>55 THEN 163
     0
1590 IF WA=7 THEN 1780
1600 W=WA-1
1610 V=VA
1620 GOTO 1720
1630 IF T<>56 THEN 169
     0
1640 IF WA=7 THEN 1780
1650 IF VA=2 THEN 1780
1660 V=VA-1
1670 W=WA-1
1680 GOTO 1720
1690 IF T<>65 THEN 123
     0
1700 CALL HCHAR(VA,WA,
64)
1710 GOTO 4410
1720 CALL GCHAR(V,W,A)
1730 IF A<>64 THEN 178
     0
1740 CALL HCHAR(VA,WA,
     88)
1750 CALL SOUND(50,-7,
     0)
1760 CALL HCHAR(V,W,72
     )
1770 GOTO 1940
1780 FOR I=1 TO 5
1790 IF A<>120 THEN 18
     40
1800 CALL COLOR(12,10,
     15)
1810 CALL SOUND(10,-3,
     5)
1820 CALL COLOR(12,5,1
     5)
1830 GOTO 1920
1840 IF A=88 THEN 1890
1850 CALL COLOR(7,13,1
     5)
1860 CALL SOUND(10,-1,
     10)
1870 CALL COLOR(7,9,15
     )
1880 GOTO 1920
1890 CALL COLOR(8,10,1
     5)
1900 CALL SOUND(10,-3,
     5)
1910 CALL COLOR(8,7,15
     )
1920 NEXT I
1930 GOTO 1230
1940 IF O<DO THEN 2130
1950 IF YA<=H THEN 198
     0
1960 IF YA>=B THEN 201
     0
1970 GOTO 2040
1980 IF XA<=G THEN 268
     0
1990 IF XA>=D THEN 259
     0
2000 GOTO 2410
2010 IF XA<=G THEN 286
     0
2020 IF XA>=D THEN 277
     0
2030 GOTO 2500
2040 IF XA<=G THEN 207
     0
2050 IF XA>=D THEN 210
     0
2060 GOTO 2130
2070 IF YA<=H THEN 268
     0
2080 IF YA>=B THEN 286
     0
2090 GOTO 2230
2100 IF YA<=H THEN 259
```

```
     0
2110 IF YA>=B THEN 277
     0
2120 GOTO 2320
2130 IF V<=YA THEN 217
     0
2140 IF W<=XA THEN 259
     0
2150 IF W>XA THEN 2680
2160 GOTO 2410
2170 IF V=YA THEN 2210
2180 IF W<XA THEN 2770
2190 IF W>=XA THEN 286
     0
2200 GOTO 2500
2210 IF W<XA THEN 2320
2220 IF W>XA THEN 2230
2230 GOSUB 3130
2240 GOSUB 3040
2250 GOSUB 3210
2260 GOSUB 2950
2270 GOSUB 3300
2280 GOSUB 3550
2290 GOSUB 3380
2300 GOSUB 3470
2310 GOTO 4110
2320 GOSUB 3470
2330 GOSUB 3380
2340 GOSUB 3550
2350 GOSUB 3300
2360 GOSUB 2950
2370 GOSUB 3210
2380 GOSUB 3040
2390 GOSUB 3130
2400 GOTO 4110
2410 GOSUB 3300
2420 GOSUB 3380
2430 GOSUB 3210
2440 GOSUB 3470
2450 GOSUB 3130
2460 GOSUB 3550
2470 GOSUB 3040
2480 GOSUB 2950
2490 GOTO 4110
2500 GOSUB 2950
2510 GOSUB 3040
2520 GOSUB 3550
2530 GOSUB 3130
2540 GOSUB 3470
2550 GOSUB 3210
2560 GOSUB 3380
2570 GOSUB 3300
2580 GOTO 4110
2590 GOSUB 3380
2600 GOSUB 3470
2610 GOSUB 3300
2620 GOSUB 3550
2630 GOSUB 3210
2640 GOSUB 2950
2650 GOSUB 3130
2660 GOSUB 3040
2670 GOTO 4110
2680 GOSUB 3210
2690 GOSUB 3130
2700 GOSUB 3300
2710 GOSUB 3380
2720 GOSUB 3040
2730 GOSUB 2950
2740 GOSUB 3470
2750 GOSUB 3550
2760 GOTO 4110
2770 GOSUB 3550
2780 GOSUB 3470
2790 GOSUB 2950
2800 GOSUB 3380
2810 GOSUB 3040
2820 GOSUB 3130
2830 GOSUB 3300
2840 GOSUB 3210
2850 GOTO 4110
2860 GOSUB 3040
2870 GOSUB 2950
2880 GOSUB 3130
2890 GOSUB 3550
2900 GOSUB 3210
2910 GOSUB 3300
2920 GOSUB 3470
2930 GOSUB 3380
2940 GOTO 4110
2950 IF YA=2 THEN 3030
2960 CALL GCHAR(YA-1,X
     A,U)
2970 IF U<>120 THEN 30
     00
2980 Z=1
2990 GOTO 3010
3000 IF U<>64 THEN 303
     0
3010 Y=YA-1
3020 GOTO 3690
3030 RETURN
3040 IF YA=2 THEN 3030
3050 IF XA=25 THEN 303
     0
3060 CALL GCHAR(YA-1,X
     A+1,U)
3070 IF U<>120 THEN 31
     00
3080 Z=1
3090 GOTO 3110
3100 IF U<>64 THEN 303
     0
3110 X=XA+1
3120 GOTO 3630
3130 IF XA=25 THEN 303
     0
3140 CALL GCHAR(YA,XA+
     1,U)
3150 IF U<>120 THEN 31
     80
3160 Z=1
3170 GOTO 3190
3180 IF U<>64 THEN 303
     0
3190 X=XA+1
3200 GOTO 3650
3210 IF YA=16 THEN 303
     0
3220 IF XA=25 THEN 303
     0
3230 CALL GCHAR(YA+1,X
     A+1,U)
3240 IF U<>120 THEN 32
     70
3250 Z=1
3260 GOTO 3280
3270 IF U<>64 THEN 303
     0
3280 X=XA+1
3290 GOTO 3670
3300 IF YA=16 THEN 303
     0
3310 CALL GCHAR(YA+1,X
     A,U)
3320 IF U<>120 THEN 33
     50
3330 Z=1
3340 GOTO 3360
3350 IF U<>64 THEN 303
     0
3360 Y=YA+1
3370 GOTO 3690
3380 IF YA=16 THEN 303
     0
3390 IF XA=7 THEN 3030
3400 CALL GCHAR(YA+1,X
     A-1,U)
3410 IF U<>120 THEN 34
     40
3420 Z=1
3430 GOTO 3450
3440 IF U<>64 THEN 303
     0
3450 X=XA-1
3460 GOTO 3670
3470 IF XA=7 THEN 3030
3480 CALL GCHAR(YA,XA-
     1,U)
3490 IF U<>120 THEN 35
     20
3500 Z=1
3510 GOTO 3530
3520 IF U<>64 THEN 303
     0
3530 X=XA-1
3540 GOTO 3650
3550 IF YA=2 THEN 3030
3560 IF XA=7 THEN 3030
3570 CALL GCHAR(YA-1,X
     A-1,U)
3580 IF U<>120 THEN 36
     10
3590 Z=1
3600 GOTO 3620
3610 IF U<>64 THEN 303
     0
```

```
3620 X=XA-1
3630 Y=YA-1
3640 GOTO 3700
3650 Y=YA
3660 GOTO 3700
3670 Y=YA+1
3680 GOTO 3700
3690 X=XA
3700 IF ND>0 THEN 3750
3710 IF P2>P1 THEN 374
     0
3720 ND=INT((25-15+1)*
     RND)+10
3730 GOTO 3750
3740 ND=INT((15-5+1)*R
     ND)+5
3750 TJ=TJ+1
3760 IF TJ<ND THEN 397
     0
3770 TJ=0
3780 ND=0
3790 DM=0
3800 OB=INT((6-1+1)*RN
     D)+1
3810 IF OB>2 THEN 3870
3820 IF P1>=P2 THEN 38
     50
3830 OB=8
3840 GOTO 3970
3850 OB=4
3860 GOTO 3970
3870 IF OB>4 THEN 3930
3880 IF P1>=P2 THEN 39
     10
3890 OB=9
3900 GOTO 3970
3910 OB=5
3920 GOTO 3970
3930 IF P1>=P2 THEN 39
     60
3940 OB=10
3950 GOTO 3970
3960 OB=6
3970 IF OB=0 THEN 4030
3980 CALL HCHAR(YA,XA,
     120)
3990 DP=DP+1
4000 IF DP<OB THEN 408
     0
4010 OB=0
4020 DP=0
4030 IF Z=0 THEN 4070
4040 CALL HCHAR(YA,XA,
     120)
4050 Z=0
4060 GOTO 4080
4070 CALL HCHAR(YA,XA,
     64)
4080 CALL SOUND(50,-5,
     0)
4090 CALL HCHAR(Y,X,80
     )
4100 GOTO 4230
4110 E=1
4120 P2=P2+1
4130 CALL HCHAR(YA,XA,
     32)
4140 FOR P=0 TO 30 STE
     P 3
4150 CALL SCREEN(E)
4160 CALL SOUND(50,-5,
     P)
4170 E=E+1
4180 NEXT P
4190 CALL SCREEN(8)
4200 MY=MY+2
4210 CALL HCHAR(MY,3,3
     4)
4220 GOTO 4780
4230 CALL GCHAR(V-1,W,
     U)
4240 IF U=64 THEN 1040
4250 CALL GCHAR(V-1,W+
     1,U)
4260 IF U=64 THEN 1040
4270 CALL GCHAR(V,W+1,
     U)
4280 IF U=64 THEN 1040
4290 CALL GCHAR(V+1,W+
     1,U)
4300 IF U=64 THEN 1040
4310 CALL GCHAR(V+1,W,
     U)
4320 IF U=64 THEN 1040
4330 CALL GCHAR(V+1,W-
     1,U)
4340 IF U=64 THEN 1040
4350 CALL GCHAR(V,W-1,
     U)
4360 IF U=64 THEN 1040
4370 CALL GCHAR(V-1,W-
     1,U)
4380 IF U=64 THEN 1040
4390 CALL HCHAR(V,W,32
     )
4400 WA=W
4410 A=73
4420 V=18
4430 FOR P=1000 TO 400
     STEP -100
4440 CALL SCREEN(16)
4450 CALL COLOR(6,2,1)
4460 CALL SOUND(1,P,10
     )
4470 IF WA<=16 THEN 45
     20
4480 CALL HCHAR(V,20,A
     )
4490 VA=V
4500 CALL HCHAR(VA,20,
     32)
4510 GOTO 4550
4520 CALL HCHAR(V,10,A
     )
4530 VA=V
4540 CALL HCHAR(VA,10,
     32)
4550 V=V+1
4560 A=A+1
4570 NEXT P
4580 IF WA>16 THEN 461
     0
4590 CALL HCHAR(24,10,
     76)
4600 GOTO 4630
4610 CALL HCHAR(24,20,
     79)
4620 VA=0
4630 V=0
4640 A=0
4650 CALL SCREEN(8)
4660 MP=MP+2
4670 CALL HCHAR(MP,30,
     33)
4680 P1=P1+1
4690 GOTO 4780
4700 FOR I=1 TO LEN(M$
     )
4710 CALL HCHAR(22,2+I
     ,ASC(SEG$(M$,I,1)
     ))
4720 CALL SOUND(1,-1,2
     0)
4730 NEXT I
4740 RETURN
4750 CALL HCHAR(22,3,3
     2,28)
4760 M$=""
4770 RETURN
4780 IF PP=10 THEN 490
     0
4790 M$="nouvelle part
     ie  o  n"
4800 GOSUB 4700
4810 CALL KEY(3,T,S)
4820 IF S=0 THEN 4810
4830 IF T<>78 THEN 485
     0
4840 GOTO 4900
4850 IF T<>79 THEN 481
     0
4860 GOSUB 4750
4870 CALL HCHAR(24,10,
     32)
4880 CALL HCHAR(24,20,
     32)
4890 GOTO 670
4900 CALL CLEAR
4910 PRINT "nombre de
     parties jouees ";
     PP
4920 PRINT :::::"
     pour vous ";P2
4930 PRINT ::"
     pour  moi ";P1:::
4940 END
```

Un nouveau module graphique pour le TI-99/4A

Henri Mathian

Un nouveau module conçu et fabriqué par MECHATRONIC, de Sindelfingen en Allemagne, est disponible en France depuis début juin. Son nom : "Le Basic Graphique Etendu Apesoft". Ses capacités : "Super !". Il apporte une nouvelle jeunesse à notre bon vieux TEXAS avec la quasi- totalité des commandes graphiques disponibles sur le micro-pomme. De plus il est accompagné d'un manuel en français. Tout pour plaire ce petit module... même son prix (1200,00 F à la Règle à Calcul !).

Cet article n'a pour but que de donner un rapide aperçu des possibilités de ce module qui possède 44 Ko de mémoire préprogrammée.

Le principe est très simple : le déplacement d'un curseur graphique génère le tracé d'une ligne. L'écran sert de feuille de travail. A l'initialisation du "mode graphique" une fenêtre est définie à l'intérieur de laquelle le curseur partant d'un point donné va, selon les instructions reçues, tracer la courbe demandée avec une grande rapidité.

L'adressage de chaque point de la fenêtre graphique devient possible, et cela en 16 couleurs, pour le premier et l'arrière plan. 40 instructions graphiques sont disponibles et peuvent être appelées sous le TI-Basic ou le TI-Basic Etendu. La composition de programmes facilement agrémentée de graphiques haute résolution devient possible. Contrairement à ce que dit la brochure en page 42, les sous-programmes CALL SOUND et CALL SPRITE peuvent être utilisés même en mode graphique. Ainsi il est possible d'ouvrir une fenêtre, de créer un graphique, de le dupliquer éventuellement sur une autre partie de l'écran ou de le faire se déplacer, de définir un ou plusieurs lutins, de les appeler, de les déplacer sur fond musical sans altérer le graphisme préalablement composé.

Ce module comporte aussi une instruction très pratique "CALL BHCOPY" qui permet de réaliser une copie d'écran en mode Bit-Map sur imprimantes à aiguilles de types GP 550, EPSON FX-80, RX-80 et compatibles. Avec toutefois une petite restriction : Les SPRITES ne sont pas imprimés.

L'utilisation de ce module nécessite obligatoirement la présence de l'extension 32 Ko.

"C'est extraordinaire de voir fonctionner le XG Basic Apesoft" dit la brochure."Ce n'est pas faux!" répond l'utilisateur.

Mode de fonctionnement

Le déplacement d'un curseur génère le tracé d'une ligne à l'intérieur d'une fenêtre graphique qui comporte 128 colonnes (16 caractères * 8 points) et 120 lignes (15 caractères * 8 points) soit 15 360 points.

Cette fenêtre peut être affichée seule en un endroit quelconque de l'écran, en partie ou totalement en dehors de l'écran, elle peut être reproduite une ou plusieurs fois sur la totalité de l'écran en juxtaposition, superposition partielle, être déplacée à la manière d'un escargot (boucle FOR-NEXT avec effacement de la fenêtre précédente), être découpée ou être réduite (affichage d'une portion de la fenêtre).

La figure exécutée dans la fenêtre graphique est unique. On ne peut ouvrir deux fenêtres à l'écran et y tracer des figures différentes. Le nouveau tracé destiné à la fenêtre 3 ira simultanément s'inscrire en superposition dans les 3 fenêtres ouvertes. C'est dommage ! A nous de trouver le truc !

La limitation à 15 360 points est nécessaire afin de réserver suffisamment de place dans la VDP-RAM pour les programmes Basic, les variables alpha-numériques, etc...

L'adressage direct de la totalité de l'écran soit 256 colonnes (32 caractères * 8 points) et 196 lignes(24 caractères * 8 points), ce qui représente 50 176 points, demande 12 Ko de la

VDP-RAM ; les tableaux de caractères et de couleurs débordent sur les tables d'adresses de l'interpréteur Basic. Si l'adressage individuel de ces 50 176 points est possible, la fenêtre ouverte à l'écran pour créer le graphique ne peut malheureusement dépasser 15 360 points dans la version actuelle du logiciel.

Quelques instructions graphiques

CALL APESOFT

Le sous-programme APESOFT ferme tous les fichiers encore ouverts, transfère les routines graphiques depuis le module vers l'extension mémoire 32 Ko, réserve la place nécessaire pour le graphique dans la VDP-RAM puis exécute une instruction NEW de façon à effacer tous les programmes se trouvant en mémoire.

Les routines graphiques sont constituées d'instructions très puissantes qui simplifient énormément la réalisation de dessins complexes.

CALL LINK("GRAFIC", MODUS)

Annonce au calculateur le mode graphique et initialise tous les registres.

CALL LINK("WINDOW", Colonne,Ligne)

Positionne la fenêtre graphique à l'écran.

CALL LINK("WINDOW",Z,S, ZA,SA,DZ,DS)

Positionne une ou plusieurs portions de la fenêtre.

CALL LINK("SETBLE", Nombre de colonnes)

Permet d'obtenir une fenêtre à dimensions variables (la surface maximale restant toujours, dans l'état actuel de mes connaissances, de 15 360 pts).

CALL LINK("SETCOL", Couleur Premier Plan, Couleur Arrière Plan)

Chacune des 16 couleurs du Basic TI peut être utilisée soit dans le premier plan, soit dans l'arrière plan. On peut utiliser simultanément dans un graphique plusieurs couleurs au premier et à l'arrière plans.

CALL LINK("INVERT",X,Y, DX,DY)

Intervertit les couleurs du premier et de l'arrière plans.

CALL LINK("CLSCRN")

Equivalent à CALL CLEAR, efface le graphique, les paramètres internes du curseur restant inchangés.

CALL LINK("CENTRE",X,Y)

Définit le système de coordonnées utilisateur.

CALL LINK ("MOVE", Distance)

Permet de tracer une ligne de longueur "Distance" à partir de la position actuelle X,Y du curseur.

CALL LINK("MOVETO",X,Y)

Trace une ligne depuis la position actuelle du curseur jusqu'au point défini par les coordonnées X et Y.

CALL LINK("TURN",PHI)

Ajoute la valeur PHI exprimée en degrés à l'angle courant du curseur.

CALL LINK("RECT",A,B)

Trace un rectangle.

CALL LINK("CIRCLE",X,Y, R)

Trace un cerle de centre(X,Y) de rayon R.

CALL LINK("ARCUS",X,Y,R, PHI,DPHI)

Trace un arc circulaire.

CALL LINK("ELLIPS",X,Y, A,B)

Trace une ellipse centrée en X,Y de demi-grand axe A, de demi-petit axe B.

CALL LINK("VALUES",X,Y, PHI,FG,BG)

Affecte aux variables désignées les valeurs actuelles des paramètres.Coordonnées curseur (X,Y). Angle curseur (PHI). Couleur premier plan (FG). Couleur de l'arrière plan.

CALL LINK("HSTDIA",X,Y, WIDTH,HEIGHT,DEPTH)

Trace une barre de diagramme.

Composition graphique

CALL LINK("WRITE",Z,S, STRING)

Permet le mélange de graphique et de texte.

CALL LINK("DSPLAY",Z,S, SIZE,VAR$)

Correspond à "DISPLAY AT".

CALL LINK("ACCEPT",Z,S, SIZE,VAR$)

Correspond à "ACCEPT AT"

Sauvegarde

CALL GSAVE("DATEINAME")

Les dessins ne peuvent être stockés que sur floppy disques.

CALL GLOAD("DATEINAME")

Cette instruction charge dans le VDP-RAM le dessin enregistré sur la disquette. L'appel de GLOAD sans avoir, au préalable, appelé "GRAFIC", entraine un message d'erreur.

Quelques instructions non graphiques

qui viennent s'ajouter à l'ensemble des commandes et instructions du TI Extended Basic que possède également ce module :

CALL BHCOPY("FILENAME", "ESC-SEQUENZ")

Réalise une copie d'écran. ex.:

CALL BHCOPY("PIO","K") produit une copie "parallèle" sur une BMC.

CALL VPEEK

Lecture des positions de mémoire dans la VDP-RAM.

CALL VPOKE

Ecriture directe dans les positions de la VDP-RAM.

CALL GPEEK

Retourne les caractères contenus dans les valeurs ASCII 32 à 126 à leurs définition de base. Très proche de CALL CHARSET.

CALL WAIT

Attente de 0 à 16 382 (1000 correspond à 20 secondes).

CALL MOVE

Déplace le contenu de blocs de mémoire dans l'ordinateur.

CALL MSAVE

Enregistrement sur cassette de programmes en langage machine.

CALL MLOAD

Charge le contenu de la mémoire vive, sauvegardé par MSAVE.

CALL BYE

Peut être utilisé en cours de programme.

CALL NEW

Peut être utilisé à l'intérieur d'un programme.

CALL RESTORE (A)

Instruction RESTORE comportant une variable.

CALL QUIT OF

Met la touche QUIT hors fonction.

CALL QUIT ON

Rétablit la fonction.

CALL SPRON

Mise en mouvement de tous les sprites au même moment.

CALL SPROF

Arrête le mouvement de tous les sprites.

CALL SCREENON

Allume l'écran.

CALL SCREENOF

Eteint l'écran.

CALL FIND("Cherchestring"," Stringarry"(),Variableretour)

Dans un alignement de Strings d'une seule dimension, le sous-programme FIND cherche la notion définie dans "Cherchestrings". La variable retour numérique prend la valeur de la notion recherchée dans l'alignement.

Cette liste d'instructions graphiques ou non graphiques n'est pas exhaustive. Elle donne une idée des possibilités surprenantes de ce petit module qui donne vraiment envie de continuer, ou de se remettre, à tapoter les touches de ce cher vieux TEXAS.

99

Le module "MAXIMEM 48Ko" pour TI-99/4A *et* TI-99/4

Ce nouveau module, qui nous a été présenté sous la forme d'un prototype, offre de nombreuse possibilités :

- il double la capacité mémoire du TI 99-4 ou 4/A ;
- Maximem se connecte à l'endroit réservé aux modules ;
- il ne nécessite aucune alimentation extérieure ;
- il permet de charger et exécuter automatiquement tout module mis préalablement sur disque ;
- Maximem peut rester indéfiniment en place, ce qui évite l'usure du port d'entrée de la console ;
- Maximem met à votre disposition plusieurs choix de modules après la mire de couleurs, par exemple : Editeur/Assembleur, Basic Etendu, Zero Zap, Basic TI.

Utilisation

Maximem est simple à utiliser : après la mire de couleurs, trois choix s'offrent à l'utilisateur :

1) TI Basic (rien n'est changé pour le moment) ;
2) MAXIMEM (qui charge le module de votre choix et lance son exécution) ;
3) Editeur/assembleur (une version améliorée du module Editor/assembler).

Caractéristiques techniques

Capacité 48Ko répartis comme suit (toutes les adresses en hexadécimal) :

- 6000 à 7FFF RAM 1 (8Ko)
- 6000 à 7FFF RAM 2 (8Ko) *(les RAM 1 et RAM 2 sont commutables par "flag")*
- 6000 à 77FF GRAM (6Ko)
- 8000 à 97FF GRAM (6Ko)
- A000 à B7FF GRAM (6Ko)
- C000 à D7FF GRAM (6Ko)
- E000 à FFFF GRAM (8Ko)

Equipement requis

L'utilisation de ce module requiert un lecteur de disquette avec son contrôleur et une extension de mémoire 32Ko.

A l'avenir, une version cassette de MAXIMEM permettra l'usage seul du magnétophone à cassettes pour les modules ne nécessitant pas l'extension mémoire.

Nous ne savons pas encore quel sera le prix de ce très intéressant module, ni par qui il sera distribué. Si vous désirez obtenir des précisions, vous pouvez contacter le fabriquant :

Gournay Guy
933, Delorimier
Longueuil
J4K 3M8
PQ Canada

99

Courrier des Lecteurs

Alexandre Duback

Je viens d'acheter la Mini-mémoire et j'ai réussi à subtiliser le livre "Initiation au langage assembleur du TI-99/4A", de Denise Amrouche et Roger Didi pour l'étudier et voilà mon problème : je suis au chapitre 5, étude de la VDP RAM.

Il est dit (page 38), à propos des lutins, que la table des descriptions de leur forme s'étend de >0400 à >07FF ; soit 1024 octets. Il y a possibilité de définir 128 lutins. On nous dit aussi (page 39), à propos des vitesses, que la table s'étend de >0780 à >07FF. Cette table des vitesses empiète donc sur les 1024 octets réservés à la définition des 128 lutins. Pourquoi ? Si ces adresses sont justes, cela veut dire que je ne peux définir que 112 lutins : >0400 à >077F (page 40).

Il y a aussi un autre problème : il est possible de définir 256 caractères, moins les trente deux premiers : >0800 à >0FFF. Nous pouvons facilement trouver leur code ASCII. Quand je définis un lutin, je lui donne un code ASCII allant généralement de 128 à 256 (Si il y en a 128 définissables) Je peut aussi définir des caractères allant de 128 à 256. Quand je demande l'apparition d'un cube à l'écran, de code ASCII 128, que celui-ci est un lutin défini dans la table des lutins et aussi défini dans la table des caractères, est-ce la forme du lutin ou celle du caractère qui va apparaître sur l'écran, ayant tous deux le même code ASCII ?

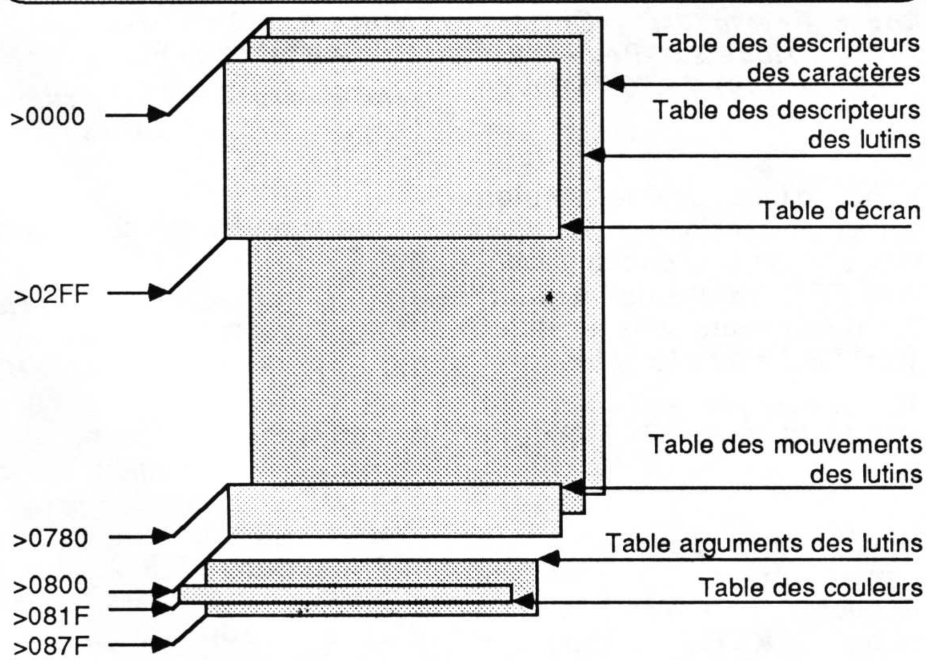
Frédéric Rejol - 42100 St Etienne

Avec la Mini-mémoire et le Basic TI, les différentes tables utilisées pour l'affichage des caractères et lutins se superposent, ceci afin de laisser de la place pour le programme Basic qui est, lui aussi, loger dans la mémoire vidéo (seule zone de mémoire vive disponible sans extension 32Ko, si l'on ne tient pas compte des 256 octets du TMS 9900). Le schéma ci-joint visualise la mémoire et explique que les possibilités se trouvent réduites par cette configuration par défaut.

Exemple de mémoire vidéo configurée depuis un programme en assembleur



Configuration par défaut de la mémoire vidéo lors de l'emploi du Basic TI



Fort heureusement, en assembleur - puisque c'est ce qui semble vous préoccuper - il est parfaitement possible de partitionner la mémoire dans le but d'obtenir des tables complètes. Le second schéma proposé montre une configuration possible parmi d'autres. Puisque ceci n'est peut-être pas évident, quelques rappels et explications s'imposent.

L'adresse de base de la majorité des tables n'est pas figée puisqu'elle est fonction de l'état de l'un des registres du processeur vidéo. Ainsi, le contenu du registre 2 fixe l'adresse de base de la table d'écran, le registre 3 fixe l'adresse de base de la table des couleurs, etc... Seule la table utilisée pour les mouvements des lutins doit impérativement se trouver en >780 (1920).

Pour fixer l'adresse de base de la table d'écran, il faut placer dans le registre 2 une valeur correspondante à l'adresse voulue divisée par >400 (1024). Dans notre exemple, l'adresse de base de la table d'écran est >0000, on place donc dans le registre 2 : >0000/>400, soit 0.

Pour la table des couleurs, même méthode mais il faut cette fois utiliser le registre 3, et diviser l'adresse de base désirée par >40 (64). Pour que l'adresse de base de la table des couleurs soit en >380 (896), il faut placer la valeur >E (14).

Le registre 4 est employé pour la table des descripteurs des caractères. La valeur à placer dans le registre doit correspondre à l'adresse de base voulue, divisée par >800 (2048). Dans le cas qui nous intéresse,et où nous avons situé la table en >800, le registre vidéo concerné doit contenir >800/>800, soit 1.

Pour la table des argument des lutins (couleurs, positions,etc...) on utilise le registre 5, dans lequel on place l'adresse de base de la table, divisée par >80 (128). Dans le cas où l'adresse de base doit être >300 (768), on place 6 dans le registre 5.

Le registre 6 définie l'adresse de base de la table des descripteurs des lutins (formes). la valeur à indiquer correspond à l'adresse de base divisée par >800 (2048). Pour une adresse de base en >1000 (4096), il faut placer dans le registre 6 la valeur 2 (2 * 2048 = 4096).

Cette méthode permet d'obtenir 256 caractères utilisables, avec 32 groupes de 8 caractères pour les couleurs, et 32 lutins avec 256 définitions différentes au même moment (ou 64 dans le cas de lutins en 16*16 pixels).

99

Petites annonces (gratuites)

Ventes

A vendre : TI-99/4A bon état (4.83), Basic étendu, câble cassettes, Mini-mémoire, joysticks et huit modules de jeux (Parsec, Star Trek, Burger Time...) plus nombreux jeux sur cassettes et livres. Le tout : 3500 FF.

Gilles Bompard - 12, place Béranger - 77170 Servon - Tél. : 405.11.66.

Vends TI-99/4A, Basic étendu, Mini-mémoire, Echecs, livres et cassettes. Très bon état. Prix à débattre.

Roger Feyssaguet - Chizé - 79170 Brioux sur Boutonne - Tél. : 16 (49) 76.70.76 après 19H.

Vend, cause double emploi, boîtier d'extension péripheriques avec câble et carte de commande : 700 FF ; extension mémoire 32Ko extérieure sous garantie : 1000 FF. Le lot : 1500 FF.

M. Lepercque - Tél. : (1) 563.12.12 poste 3748 (heures bureau).

Vends boîtier d'extension avec contrôleur et lecteur de disquettes, carte mémoire 32Ko et carte RS232 : 7000 FF. Editeur/assembleur : 500 FF. TI Writer Word Processor : 800 FF. TI LOGO II : 800 FF.

Jean-Pierre IVES - 5, rue de l'Argile - 34160 Castries - Tél. : (67) 70.07.74.

Vends synthétiseur de paroles : 400 FF.

J-F. de Belleval - La Borne Trouée - 60153 Rethondes - Tél. : (4) 485.62.26 après 18H.

Vends TI-99/4A, câble magnéto, joysticks, Basic étendu, Parsec, TI Invaders, Lunar Jumper, Car Wars, Gestion de fichiers, livres, "99 Magazine" 1 à 7, plus de nombreux programmes. Valeur : plus de 3500 FF ; vendu 2500 FF.

Olivier Serreault - Plan de Leydet - 04200 Sisteron.

Vends nombreuses cassettes en Basic TI et Basic étendu ("99 Magazine" et "Hebdogiciel") pas cher et en bon état.

Pierre-E. Gougelet - 28, rue Guillaumet - 51400 Livry-Louvercy.

Vends TI-99/4A Péritel, Basic étendu Français, TI LOGO II, mémoire 32Ko externe, interface parallèle externe, magnéto TI, deux câbles cassettes, numéros 1 à 8 de "99 Magazine" avec cassettes d'accompagnement, modules Echecs, Gestion privée, Gestion de fichiers, manettes de jeux TI, cassette "Basic par soi-même", livres ("Boîte à outils", "A l'affiche", "102 programmes", "Jeux, trucs et comptes").

Serge Nardizzi - résidence Les Jardins de l'Aude - Batiment E - 11000 Carcassonne - Tél. : (68) 47.59.82 après 20H30.

Vends boîtier d'extension, carte mémoire 32Ko, carte RS232 (sorties série et parallèle), contrôleur et lecteur de disquettes, imprimante Star Gemini 10 (120 cps, avec cordon), synthétiseur vocal, modules Basic étendu, TI-Writer, TI LOGO 2, Gestion de fichiers, Gestion privée et TI Calc.

Michel Viaene - Longueville sue Aube - 10170 Méry sur Seine - Tél. : (25) 21.22.32.

Vends modules Tombstone, Video Games I, Wumpus, Star Trek, Mash, Car Wars, Chilshom Trail, A maze ing : 100 FF chacun. Cassettes Rétro I et II, Lunar Lander, Lunar Jumper, Solar System : 50 FF chaque.

J. Reibel - 9, square V. Fleming - 92350 Le Plessis Robinson - Tél. : 16 (1) 631.46.11.

Vends modules Basic étendu, Hunt the Wumpus, et les cassettes Lunar Jumper, Treasure Trap et Pilot. Tous avec manuels, vente ensemble ou séparément.

M. Heudes - 17, rue de Léningrad - 75008 Paris - Tél. : 522.78.48.

Vends modules TI Invaders, Football, Blasto, Munch Man : 120 FF pièce ; Pole Position, Parsec : 160 FF pièce. Vends "99 Magazine" numéros 1 à 8 : 20 FF le numéro. Le tout est en très bon état.

T. Martel - 12, rue Bonne Santé - 76620 Le Havre - Tél. : (35) 45.30.58.

Vends TI-99/4A, poignées de jeux, câble cassettes, deux livres, TI Invaders, Hustle, Star Trek, Music Maker, huit cassettes "99 Magazine". Valeur 2800 FF ; vendu 1400 FF.

P. Petitjean - 7, rue d'Alembert - 18000 Bourges - Tél. : (48) 65.62.54.

Vends TI-99/4A avec Basic étendu en Français, cassette "Basic par soi-même", autres cassettes et livres de programmes : 2000 FF.

J-M. Lesage - Pré Bercy III - 03000 Moulins - Tél. : (70) 20.13.84.

Achats

Achète disquette "TI Writer".

G. Crittin - 14 bis, chemin du Bochet - 1196 Gland (Suisse) - Tél. : 19.41.22.64.38.50.

Achète contrôleur de disquettes seul 1500 FF ou avec lecteur de disquettes simple face 3500 FF (à débattre). Faire offre à :

Pascal Thomas - Paris - Tél. : (1) 203.71.10.

Achète modules TI Calc, Multiplan, Stat, Echecs, Maths (Collins), etc... Interface parallèle ou série pour imprimante et console, même en panne.

M. Voisin - 22, route de Vewerque - Grépia - Tél. : (61) 08.21.88.

Achète module Mini-mémoire complet (manuel d'utilisation).

Jean-Louis Philippe - 73, route de Sierck - Koenigsmaker - 57110 Yutz - Tél. : 16 (8) 250.01.46.

Cherche mode d'emploi de la Mini-mémoire, Modules TI Writer, Statistiques, Physical Fitness, Numération I & II, Magie des nombres, disquette Dictée magique, plans et schémas de l'interface RS232C (offre possibilité réalisations) et/ou extension mémoire 32Ko.

Philippe Mechelaere - 6, rue des Saints Sauveurs - 92260 Fontenay aux Roses - Tél. : 16 (1) 350.88.18 après 20H (ou laisser message).

Achète module "Speech Editor" (200 à 300 FF). Faire offre à :

M. Couegnat - 66, rue du Repas - 69007 Lyon - Tél. : 16 (1) 858.21.16.

Cherche carte interface RS232C pour boîtier d'extension, module TI Calc, modules et disquettes Multiplan et TI Writer, console TI-99/4A et accessoires hors service pour bricolages.

Henri Espi - résidence Le Poséïdon - 66750 Saint Cyprien Plage - Tél. : (68) 21.22.12 après 19H.

Recherche Modules "Speech Editor" et "LOGO 2". Carte mémoire 32Ko, lecteur et contrôleur de disquettes internes. Etudie toutes propositions.

Guy Mahé - 23, allée H. Sellier - A.D 1156 - 92800 Puteaux.

Divers

Echange manuel "Editor/Assembler". Faire offre à :

Gilbert Vigliano - Lot "La Plage" - Batiment A - avenue Jean d'Agrèves - 83400 Hyères - Tél. : (94) 57.20.74.

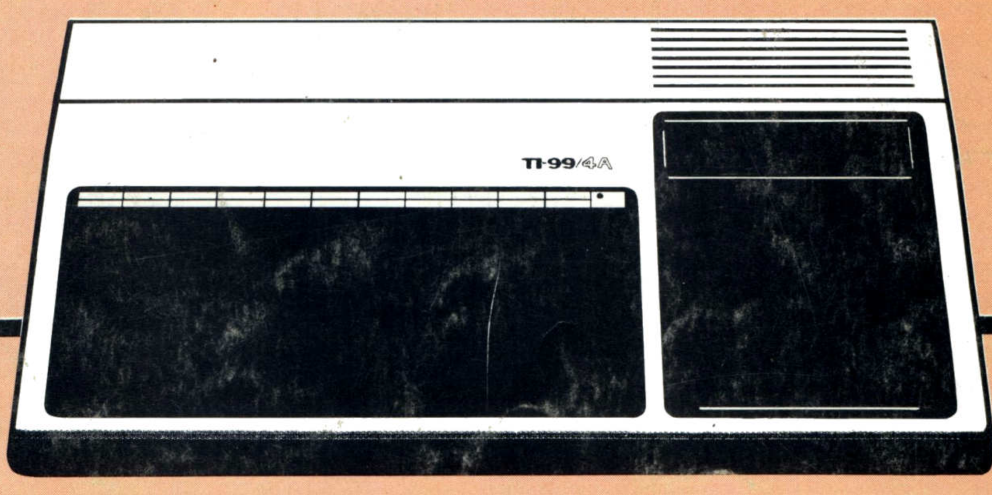
Recherche schémas complets de l'interface RS232 du TI-99, ainsi que son mode d'emploi.

Jean-Jacques Graff - 23, rue du Ballon d'Alsace - 68740 Fessenheim - Tél. : (89) 48.61.37.

99 MAGAZINE

CASSETTE D'ACCOMPAGNEMENT DISPONIBLE

Un puzzle de chiffres
Expérience pédagogique
Transfert rapide d'informations
Tracé de courbes par splines cubiques



Numéro 10 - Trimestriel - Septembre 1985 - 40 F

ISSN 0759-9005

Sommaires des numéros 1 à 9 de "99 Magazine"

Numéro 1

Musik Maker à l'essai	
Pourquoi acheter un TI-99/4A	
A vous de programmer	
Déballage du TI-99	
Les avantages du Basic étendu	BE
Initiation au Basic	BTI
Jeu du Pendu	BTI
Fabriquez votre joystick	
Appelez moi KEY	BTI/BE
Programmes d'aide à la décision	BE
Jeu de l'Alphabet	BTI
Le club d'utilisateurs TI-Education	
Introduction au LOGO	L
Les tours de Hanoï	BTI
Récupérez les programmes Apple	BTI/BE
Le troisième menuet de Bach	BTI
Le dernier robot	BTI
Cherchez l'ancêtre	BTI
Gestion personnelle de fichiers	
Quand le TI devient un harmonium	BTI
Fichiers en Basic	BE

Numéro 2

Micro-monde LOGO : les lutins	L
La bataille navale	BTI
Le crocodile savant	
Jeu du Simon	BTI/BE
Jeu des carrés	BTI
L'ordinateur de l'avenir	
TI-Writer à l'essai	
Canon de Purcell	BE
Chicken Helper	BE
Sauvegarde des fichiers et prog.	BTI/BE
Le Mastermind	BTI
Othello	BTI
Astéroïdes	BE
Kermit la grenouille	BE
Le Biorythme	BTI

Numéro 3

Un programme de calcul	BE
Annonce de Texas Instruments	
Othello à l'essai	
Apprendre Basic en programmant (1)	BTI
Introduction au Pascal	P
Puissance 4	BE
Les modes d'adressage en Assembleur	A
Géographie : la carte de France	BTI
Créez votre touche RESET	
Puzzle	BTI
Awari	BTI
Le sapin de Noël	BE
PARSEC à l'essai	
LOGO, seulement pour enfant ?	L
Le Mastermind en assembleur	A
Du nouveau sur les cassettes	
Le magnétophone TI	

Numéro 4

SOS Hélicoptère	BE
Utilisations de la Mini-mémoire	BTI
Communiqué de Texas Instruments	
Le serveur de restaurant	BTI
"Echecs" à l'essai	
Combat de chars	BTI
Réalisez votre cordon de magnéto	
Apprendre Basic en programmant (2)	BTI
Jeu de mémorisation	BTI
Jack-pot	BE
Les nombres en LOGO	L
Graphisme en haute résolution	BTI + A
Sous-programmes en assembleur	A
Horloge	BE
Un détecteur de lumière pour le TI	
Le Club M.T.I.	
Le système P-UCSD	P
Les lutins du Basic étendu	BE
Le solitaire	BTI
Claustrophobia	BTI
"DISPLAY" et "ACCEPT" avec PRK	BTI
Une erreur dans le manuel MINIMEM	
Réparez votre assembleur ligne par ligne	A

Numéro 5

Conversions	BTI
Systèmes linéaires	BTI
Structure des disques du P-Système	P
Poèmes et graphismes	BTI
Un désassembleur en assembleur	A
Les tours de Hanoï	BE
Trouvez le bon mot	BTI
Le jeu de la vie	BTI ou A
Programmathèque	
Les opérateurs logiques	BE
Apprendre Basic en programmant (3)	BTI
Bibliographie	
Warangal	BTI
Graphisme en haute résolution	BTI + A
Le trésor de l'Armada	BE
Ecriture littérale d'un nombre	P
Copie d'écran sur imprimante Epson	BTI
Description du port d'entrées/sorties	
Translet	BE
Le Club Ordinabis	
Le Club Ticazur	

Numéro 6

Des chiffres et des lettres	BTI
A l'heure du Pascal	P
Kong	BE
Mini-mémoire moins "mini"	
Edipage	BTI
Caractères ASCII/page graphique	BTI + A
Musik	BTI
Copie d'écran sur Seïkosha GP-100	BE
Nouvelles fonctions avec MMEM	BTI + A
Le Basic à la loupe	BTI ou BE
Star Treck	BE
Ecriture directe dans registres VDP	BTI
My tailor was rich	BTI
Le code de la route	BE
Un double face sur votre TI	

Numéro 7

La défausse	BTI
Le tableur TI-Calc	
Sauvegarde d'images graphiques	A
Copie d'écran	A
Copie d'écran sur Seïkosha GP-250	BE
La compilation séparée	P
Les bibliothèques	P
Trouvez votre chemin dans les jeux d'aventure TI	A
Srand	BTI
Galaxia	A
LOGO et la récursivité	L
Dessinez avec le Basic Etendu	BE
Accès au mode BitMap en Basic TI	BTI
Music Box Dancer	BTI
Des nouvelles du Club Ticazur	
Routines graphiques pour le BE	BE + A

Numéro 8

Bit-Map et Mini-mémoire	BTI + A
Le train sifflera trois fois...	BE
Comptes familiaux	BE
Mission spéciale	BE
Crayon optique	A
Le traitement des fichiers en Pascal	P
Editeur de caractères	BE
Kaléïdoscope	BE
Opérations	BTI

Numéro 9

Transfert image/cassette	BTI + A
Ticrok	TI
Pointeurs et variables en Pascal	P
Gestion de fichier en Basic étendu et en assembleur	BE ou A
Un nouveau module graphique	
Musique et Mini-mémoire	BTI + A
Le module Maximem	
Pentaxe	BTI

* **BTI** : Basic TI. **BE** : Basic étendu. **A** : Assembleur. **P** : Pascal. **L** : Logo.

Tous les numéros sont disponibles.

99 Magazine n° 10

Editions MEV – 64/70, rue des Chantiers – 78000 Versailles Directeur de la publication : Hervé Thiriez

Sommaire

Editorial	*Hervé Thiriez*	5
Un puzzle de chiffres	*Georges Goument*	6
Quatre nouvelles routines pour le crayon optique	*Bernard Bezuel*	12
Tracé de courbes en haute résolution par splines cubiques	*Gérard Santraille*	27
Expérience pédagogique : orthographe et informatique	*Philippe Misandeau*	36
Barmélo(s)	*Georges Goument*	41
La disquette "Fichier d'adresses" Texas Instruments	*Jean-Bernard Morelle*	49
Des caractères dans la page graphique haute résolution	*Gérard Baroni et Jean Marin*	51
Transfert de jeux d'aventures de cassette sur disquette	*Denise Amrouche*	54
Transfert de cassette à disquette des jeux de Scott Adams	*Michel Teyssou*	55
Transfert rapide d'informations sur cassettes	*Alexandre Amortila*	57
Courrier des lecteurs		61
Petites annonces		62

Bon de commande

Cassettes de jeux

99A	Mineur, Prêt, Electronique		à	35,00 F	
99B	Yahtzee, Chardef, Division, Régression linéaire		à	35,00 F	
99C	Dames, Nim, Division 2, Isola, Schmoo, Robots		à	35,00 F	
99D	Conjugaisons, Caractérologie, Car Driver, Poker, Cannibales		à	35,00 F	
99E	Mic-Math, Course de chavaux, Poursuite, Jeu de dés, Guerre atomique, Course en ligne droite		à	35,00 F	

Disquettes de jeux

Mastermind (cf. 99 Magazine n° 3)		à	55,00 F	
Galaxia (cf. 99 Magazine n° 7)		à	55,00 F	

Disquettes utilitaires

Routines graphiques (cf. 99 Magazine n° 4, 5, 6 et 7)		à	55,00 F	
Désassembleur (cf. 99 Magazine n° 5)		à	55,00 F	
Crayon optique (cf. 99 Magazine n° 8)		à	55,00 F	

Anciens numéros

99 Magazine **1 2 3 4 5 6 7 8 9**		à	25,00 F	
Cassette d'accompagnement **1 2 3 4 5 6 7 8 9**		à	35,00 F	

Numéros 10 et 11

99 Magazine **10 11**		à	40,00 F	
Cassette d'accompagnement **10 11**		à	55,00 F	

Total TTC

Supplément pour port avion hors CEE
15,00 F par revue et/ou cassette et/ou disquette

Montant du règlement joint

Nom : ..

Adresse : ...

Envoyer ce bon et votre règlement à :

Editions MEV - 64, rue des Chantiers - 78000 Versailles

Editorial

Lorsque, en décembre 83, nous apprîmes l'arrêt de fabrication du TI-99/4A, nous savions qu'il nous faudrait cesser prématurément l'édition de votre magazine. Cette date est proche puisque le prochain numéro de "99 Magazine" sera le dernier, le nombre de lecteurs s'étant trop amenuisé pour que nous puissons nous maintenir au-dessus du seuil de rentabilité. Les difficultés pour trouver des extensions, l'apparition du "standard" MSX et de machines à la fois bon marché et performantes ont eu raison de la fidélité à Texas Instruments de nombre de nos lecteurs.

Nous ne prenons donc plus les commandes d'abonnements dont le dernier numéro à reçevoir serait supérieur à 11. En revanche, il est toujours possible de prendre un abonnement "rétroactif".

Si votre abonnement en cours - avec ou sans cassettes - comporte des numéros au delà du 11, nous vous rembourseront de la différence d'une manière qui reste à déterminer. L'éditorial du prochain numéro vous en dira plus sur ce point. S'il vous manque des anciens numéros, vous pouvez désormais les obtenir avec une remise de 36% ; c'est le moment de compléter votre collection, d'autant plus que ceci est aussi valable pour les cassettes.

***Georges Goument**, qui nous proposait "Starline", "Ticrok" et "Pentaxe" dans le précédent numéro, a encore écrit en Basic TI deux excellents programmes : "Barmélo" et "Puzzle de chiffres" ; il s'agit de deux "casse-têtes" informatiques. On retrouve aussi dans ce numéro **Bernard Bezuel** qui nous communique quatre nouvelles routines pour son crayon optique.*
*"Tracé de courbes par splines cubique"... mystérieux, non ? **Gérard Santraille** vous dévoile tous les secrets de cette méthode, exemple à l'appui.*
***Philippe Misandeau**, qui nous avait par le passé gratifié d'une contribution, nous livre cette fois le fruit d'une expérience pédagogique livrée en tant que professeur. Toujours au chapitre des expériences, **Jean-Bernard Morelle** nous fait part de ses impressions comme utilisateur du programme "Fichier d'adresses" Texas Instruments.*
***Gérard Baroni** et **Jean Marin** (est-il encore utile de dire qu'il s'agit du club Ticazur ?) ont exploré les GROM du TI-99 pour y découvrir la définition des caractères ASCII ; bien utile lorsque l'on est en mode Bitmap.*
*Nous allons maintenant parler de transfert d'informations. **Denise Amrouche** nous propose un programme qui autorise le transfert des jeux d'aventures depuis les cassettes vers des disquettes alors que **Michel Teyssou** nous donne la possibilité de faire la même chose avec une méthode - pas un programme - moins souple, mais plus universelle.*
*Nous ne terminerons pas sans citer une contribution de haut niveau d'**Alexandre Amortila**, qui autorise des transferts d'informations cassette/mémoire et mémoire/cassette trois fois plus rapide que de coutume.*

Rédacteur en chef - Directeur de la publication : Hervé Thiriez. **Rédaction** : Jean-Luc Bazanegue, Gérard Santraile. **Dessins** : Laurent Bidot. **Ont collaboré à ce numéro** : Alexandre Amortila, Denise Amrouche, Gérard Baroni, Jean-Luc Bazanegue, Bernard Bezuel, Alexandre Duback, Georges Goument, Jean Marin, Philippe Misandeau, Jean-Bernard Morelle, Gérard Santraille, Michel Teyssou.

Editions MEV - 64/70, rue des Chantiers - 78000 Versailles - Tél. : 39.51.24.43.

Publicité : consultez les Editions MEV.

Impression : Imprimerie du Lion - Repro-Versailles - 25, rue Michel Lecomte - 75003 Paris - Tél. : 42.72.96.19.

Un puzzle de chiffres

Georges Goument

2 précautions indispensables avant d'entamer la partie : s'installer confortablement, garder à disposition eau et calmants. En effet, deux opérations aussi simples que l'addition et la soustraction pourraient bien vous mener au bord de la dépression.

Le but du jeu...

- L'ordinateur choisit un chiffre, 1 ou 0, l'affiche au dessus du jeu et remplit les 36 cases du plateau avec ce même chiffre.
- Il effectue ensuite un certain nombre de brouillages qui modifient tout ou partie de ce tableau.
- Vous sortirez vainqueur de ce jeu en remettant le chiffre d'origine dans chacune des cases.

Mode d'emploi

Le chiffre servant de base à chaque combinaison apparait en une couleur inversée à l'écran. Vous vous déplacez d'une case à l'autre avec les touches **E**, **X**, **S** et **D** (les flèches indiquant le sens du déplacement). Chacune de ces évolutions modifie en conséquence la couleur du chiffre concerné.

Vous appuierez sur ENTER pour permutter le chiffre choisi ainsi que ceux qui lui sont contigus (de deux à quatre selon la position sur le plateau).

Chaque permutation donne le résultat suivant : le chiffre sélectionné diminue de 1, les 2, 3 ou 4 chiffres contigus augmentent de 1.

Dès que vous avez reconstitués les 36 chiffres, appuyez sur **V**.

Pour éviter de trop malmener vos nerfs, vous disposerez de la touche **I**.

Les chiffres sont classés dans l'ordre suivant : 1 2 3 4 5 6 7 8 9 0 ; si vous dépassez 1, vous allez à 0 et réciproquement, au delà de 0, vous retournez à 1.

Le programme

Quelques points de repères dans le basic :

23-61
Présentation et accompagnempent musical

62-121
Redéfinition des caractères, dessin du plateau

122-154
Choix d'un chiffre par le TI

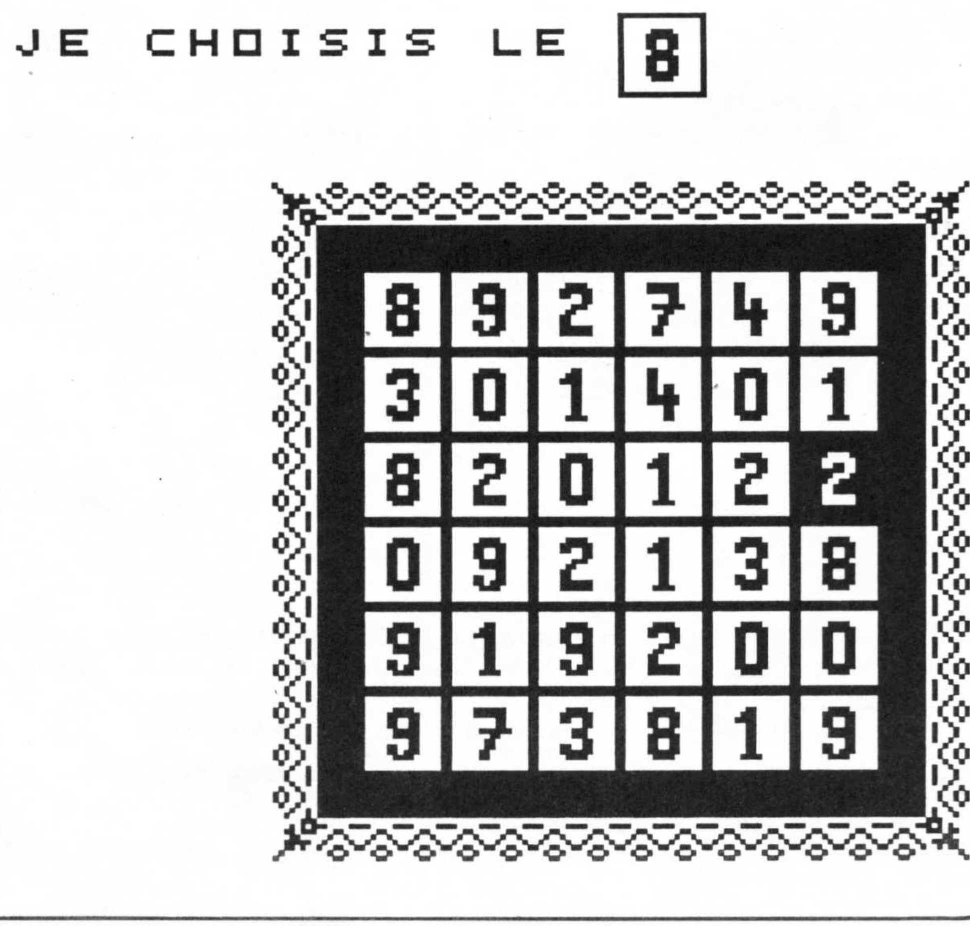
155-162
Affichage de ce chiffre au-dessus du plateau

163-182
Remplissage du plateau (36 cases). Ces lignes servent également en fin de partie avec la touche **V** pour vérifier la concordance du plateau avec la solution

183-211
Choix du nombre de brouillages (entre 16 et 36) et tirage aléatoire des chiffres à brouiller

212-222
Affichage et effacement de textes divers

223-298
Mise en attente des différentes touches du clavier



299-452
Déroulement du jeu (déplacements et permutations selon valeur du chiffre)

453-490
Sous-programmes divers : vérification, annulation, textes. Fin du jeu et commentaires.

491-503
Permet de commencer une nouvelle partie

A vous de jouer…

99

Puzzle de chiffres

```
10 REM **************
11 REM *            *
12 REM *   PUZZLE   *
13 REM *     DE     *
14 REM *  CHIFFRES  *
15 REM *            *
16 REM *  Basic TI  *
17 REM *            *
18 REM * G. Goument *
19 REM *            *
20 REM *99  Magazine*
21 REM *            *
22 REM **************
23 CALL CLEAR
24 CJ=0
25 CALL SCREEN(13)
26 FOR I=1 TO 16
27 CALL COLOR(I,1,1)
28 NEXT I
29 FOR I=40 TO 47
30 READ A$
31 CALL CHAR(I,A$)
32 NEXT I
33 DATA 8040343E18371507,01022C7C1
   8ECA8E,071537183E34408,E0A8EC18
   7C2C0201
34 DATA 182499422418C3,00C31824429
   92418,4448122525124844,221248A4
   A4481222
35 CALL CHAR(33,"FFFFFFFFFFFFFFFF"
   )
36 PRINT TAB(9);"(,,,,,,,,,,)":TAB
   (9);"/!!!!!!!!!!.":TAB(9);"/!!!
   !!!!!!!.":TAB(9);"/!!PUZZLE!!."
37 PRINT TAB(9);"/!!!!!!!!!!.":TAB
   (9);"/!!!!!!!!!!.":TAB(9);"/!CH
   IFFRES!.":TAB(9);"/!!!!!!!!!!."
38 PRINT TAB(9);"/!!!!!!!!!!.":TAB
   (9);"*----------+":::TAB(7);"je
   u de reflexion"::TAB(13);"pour"
   ::
39 PRINT TAB(11);"1 joueur"::::
40 CALL COLOR(1,14,1)
41 CALL COLOR(2,14,1)
42 CALL COLOR(3,12,1)
43 CALL COLOR(4,12,1)
44 FOR I=5 TO 8
45 CALL COLOR(I,10,14)
46 NEXT I
47 FOR I=9 TO 12
48 CALL COLOR(I,12,1)
49 NEXT I
50 GOSUB 52
51 GOTO 62
52 RESTORE 58
53 FOR I=1 TO 54
54 READ D,N
55 CALL SOUND(-50*D,N*6,4,N*4,6,N/
   2,8)
56 NEXT I
57 RETURN
58 DATA 2,523,2,587,2,587,2,587,2,
   587,2,523,4,622,2,587,2,523,2,6
   59,2,784,2,1047,4,1047,2,1047,2
   ,880
59 DATA 2,784,2,698,2,1047,2,698,2
   ,659,2,659,2,587,2,698,2,659,4,
   587,2,587,2,523,2,587,2,659,2,7
   84
60 DATA 2,784,2,659,4,784,2,784,2,
   698,2,440,2,440,2,349,3,440,2,4
   40,2,262,2,294,2,330,2,392,2,39
   2
61 DATA 2,330,4,392,2,392,2,349,2,
   349,2,220,2,220,3,349,7,220
62 CALL CLEAR
63 RANDOMIZE
64 AN=0
65 PG=0
66 VE=0
67 FOR I=1 TO 8
68 CALL COLOR(I,1,1)
69 NEXT I
70 M$="patientez je dessine le jeu
   "
71 GOSUB 212
72 FOR I=48 TO 95
73 READ A$
74 CALL CHAR(I,A$)
75 NEXT I
76 DATA FF80808183878181,FF0101818
   1818181,81818181838080FF,818181
   81C10101FF
77 DATA FFFFFFFEFCF8FEFE,FFFFFF7F7
   F7F7F7F,FEFEFEFEFCFFFFFF,7F7F7F
   7F3FFFFFFF
78 DATA FF80808387868081,FF0101C1E
   16161C1,83868687878080FF,010101
   E1E10101FF
79 DATA FFFFFFFCF8F9FFFE,FFFFFF3F1
```

```
    F9F9F3F,FCF9F9F8F8FFFFFF,FFFFFF
    1F1FFFFFF
80 DATA FF80808387868081,FF0101C1E
    16161C1,81808687838080FF,C16161
    E1C10101FF
81 DATA FFFFFFFCF8F9FFFE,FFFFFF3F1
    F9F9F3F,FEFFF9F8FCFFFFFF,3F9F9F
    1F3FFFFFF
82 DATA FF80808686868686,FF0101010
    101C1C1,87878080808080FF,E1E1C1
    C1C10101FF
83 DATA FFFFFFF9F9F9F9F9,FFFFFFFFF
    FFF3F3F,F8F8FFFFFFFFFFFF,1F1F3F
    3F3FFFFFF
84 DATA FF80808787868687,FF0101E1E
    10101C1,81808687838080FF,E16161
    E1C10101FF
85 DATA FFFFFFF8F8F9F9F8,FFFFFF1F1
    FFFFF3F,FEFFF9F8FCFFFFFF,1F9F9F
    1F3FFFFFF
86 DATA FF80808387868687,FF0101C1E
    16101C1,87868687838080FF,E16161
    E1C10101FF
87 DATA FFFFFFFCF8F9F9F8,FFFFFF3F1
    F9FFF3F,F8F9F9F8FCFFFFFF,1F9F9F
    1F3FFFFFF
88 FOR I=128 TO 159
89 READ A$
90 CALL CHAR(I,A$)
91 NEXT I
92 DATA FF80808787808083,FF0101E1E
    16161F1,80818183838080FF,C18181
    01010101FF
93 DATA FFFFFFF8F8FFFFFC,FFFFFF1F1
    F9F9F0F,FFFEFEFCFCFFFFFF,3F7F7F
    FFFFFFFFF
94 DATA FF80808387868683,FF0101C1E
    16161C1,87868687838080FF,E16161
    E1C10101FF
95 DATA FFFFFFFCF8F9F9FC,FFFFFF3F1
    F9F9F3F,F8F9F9F8FCFFFFFF,1F9F9F
    1F3FFFFFF
96 DATA FF80808387868683,FF0101C1E
    16161E1,83808686838080FF,E16161
    E1C10101FF
97 DATA FFFFFFFCF8F9F9FC,FFFFFF3F1
    F9F9F1F,FCFFF9F8FCFFFFFF,1F9F9F
    1F3FFFFFF
98 DATA FF80808387868686,FF0101C1E
    1616161,86868687838080FF,616161
    E1C10101FF
99 DATA FFFFFFFCF8F9F9F9,FFFFFF3F1
    F9F9F9F,F9F9F9F8FCFFFFFF,9F9F9F
    1F3FFFFFF
100 FOR I=1 TO 8
101 READ A,B,C,D
102 CALL HCHAR(A,B,C,D)
103 DATA 6,10,33,14,19,10,33,14,5,
    9,40,1,5,24,41,1,20,9,42,1,20,2
    4,43,1,5,10,44,14,20,10,45,14
104 NEXT I
105 CALL VCHAR(7,10,33,12)
106 CALL VCHAR(7,23,33,12)
107 CALL VCHAR(6,9,47,14)
108 CALL VCHAR(6,24,46,14)
109 CALL COLOR(3,2,5)
110 CALL COLOR(4,2,6)
111 CALL COLOR(5,2,9)
112 CALL COLOR(6,2,10)
113 CALL COLOR(7,2,11)
114 CALL COLOR(8,2,12)
115 CALL COLOR(13,2,3)
116 CALL COLOR(14,2,4)
117 CALL COLOR(15,2,8)
118 CALL COLOR(16,2,15)
119 CALL COLOR(1,14,1)
120 CALL COLOR(2,14,1)
121 GOSUB 221
122 A=INT((10-1+1)*RND)+1
123 IF A>1 THEN 126
124 A=48
125 GOTO 151
126 IF A>2 THEN 129
127 A=56
128 GOTO 151
129 IF A>3 THEN 132
130 A=64
131 GOTO 151
132 IF A>4 THEN 135
133 A=72
134 GOTO 151
135 IF A>5 THEN 138
136 A=80
137 GOTO 151
138 IF A>6 THEN 141
139 A=88
140 GOTO 151
141 IF A>7 THEN 144
142 A=128
143 GOTO 151
144 IF A>8 THEN 147
145 A=136
146 GOTO 151
147 IF A>9 THEN 150
148 A=144
149 GOTO 151
150 A=156
151 M$="je choisi le"
152 LI=1
153 CO=2
154 GOSUB 217
155 X=16
156 Y=1
157 W=A
158 FOR I=X TO X+1
159 CALL HCHAR(Y,I,A)
160 CALL HCHAR(Y+1,I,A+2)
161 A=A+1
```

```
162 NEXT I
163 M$="remplissage du plateau"
164 GOSUB 212
165 Y=7
166 FOR I=1 TO 6
167 FOR X=11 TO 21 STEP 2
168 IF VE=0 THEN 172
169 CALL GCHAR(Y,X,A)
170 IF A<>W THEN 346
171 GOTO 176
172 CALL HCHAR(Y,X,W)
173 CALL HCHAR(Y+1,X,W+2)
174 CALL HCHAR(Y,X+1,W+1)
175 CALL HCHAR(Y+1,X+1,W+3)
176 NEXT X
177 Y=Y+2
178 IF VE=0 THEN 180
179 CALL SOUND(11,1946,10)
180 NEXT I
181 IF VE=1 THEN 291
182 GOSUB 221
183 M$="brouillage des chiffres"
184 GOSUB 212
185 NP=INT((36-16+1)*RND)+16
186 FOR P=1 TO NP
187 Y=0
188 Y=INT((17-7+1)*RND)+7
189 IF Y=7 THEN 197
190 IF Y=9 THEN 197
191 IF Y=11 THEN 197
192 IF Y=13 THEN 197
193 IF Y=15 THEN 197
194 IF Y=17 THEN 197
195 GOTO 187
196 X=0
197 X=INT((21-11+1)*RND)+11
198 IF X=11 THEN 299
199 IF X=13 THEN 299
200 IF X=15 THEN 299
201 IF X=17 THEN 299
202 IF X=19 THEN 299
203 IF X=21 THEN 299
204 GOTO 196
205 NEXT P
206 GOSUB 221
207 P=37
208 M$="termine   a vous de jouer"
209 GOSUB 212
210 GOSUB 221
211 GOTO 223
212 LI=24
213 CO=2
214 FOR I=1 TO 5
215 CALL SOUND(10,-1,10)
216 NEXT I
217 FOR I=1 TO LEN(M$)
218 CALL HCHAR(LI,CO+I,ASC(SEG$(M$
   ,I,1)))
219 NEXT I
220 RETURN
221 CALL HCHAR(24,3,32,28)
222 RETURN
223 Y=11
224 X=15
225 A=0
226 C=0
227 G=0
228 U=0
229 V=0
230 DY=2
231 DX=2
232 GOSUB 299
233 YA=Y
234 XA=X
235 CALL KEY(3,T,S)
236 IF S=0 THEN 235
237 IF T<>13 THEN 255
238 CJ=CJ+1
239 IF CJ<>100 THEN 243
240 M$="pensez vous y arriver"
241 GOSUB 212
242 GOTO 250
243 IF CJ<>200 THEN 247
244 M$="ca devient assez inquietan
   t"
245 GOSUB 212
246 GOTO 250
247 IF CJ<400 THEN 390
248 M$="ras la memoire je stoppe"
249 GOSUB 212
250 FOR I=1 TO 150
251 NEXT I
252 GOSUB 221
253 IF CJ=450 THEN 488
254 GOTO 390
255 IF T<>68 THEN 263
256 GOSUB 355
257 IF XA+DX>21 THEN 261
258 X=XA+DX
259 Y=YA
260 GOTO 232
261 X=21
262 GOTO 232
263 IF T<>69 THEN 271
264 GOSUB 355
265 IF YA-DY<7 THEN 269
266 Y=YA-DY
267 X=XA
268 GOTO 232
269 Y=7
270 GOTO 232
271 IF T<>83 THEN 279
272 GOSUB 355
273 IF XA-DX<11 THEN 277
274 X=XA-DX
275 Y=YA
276 GOTO 232
277 X=11
```

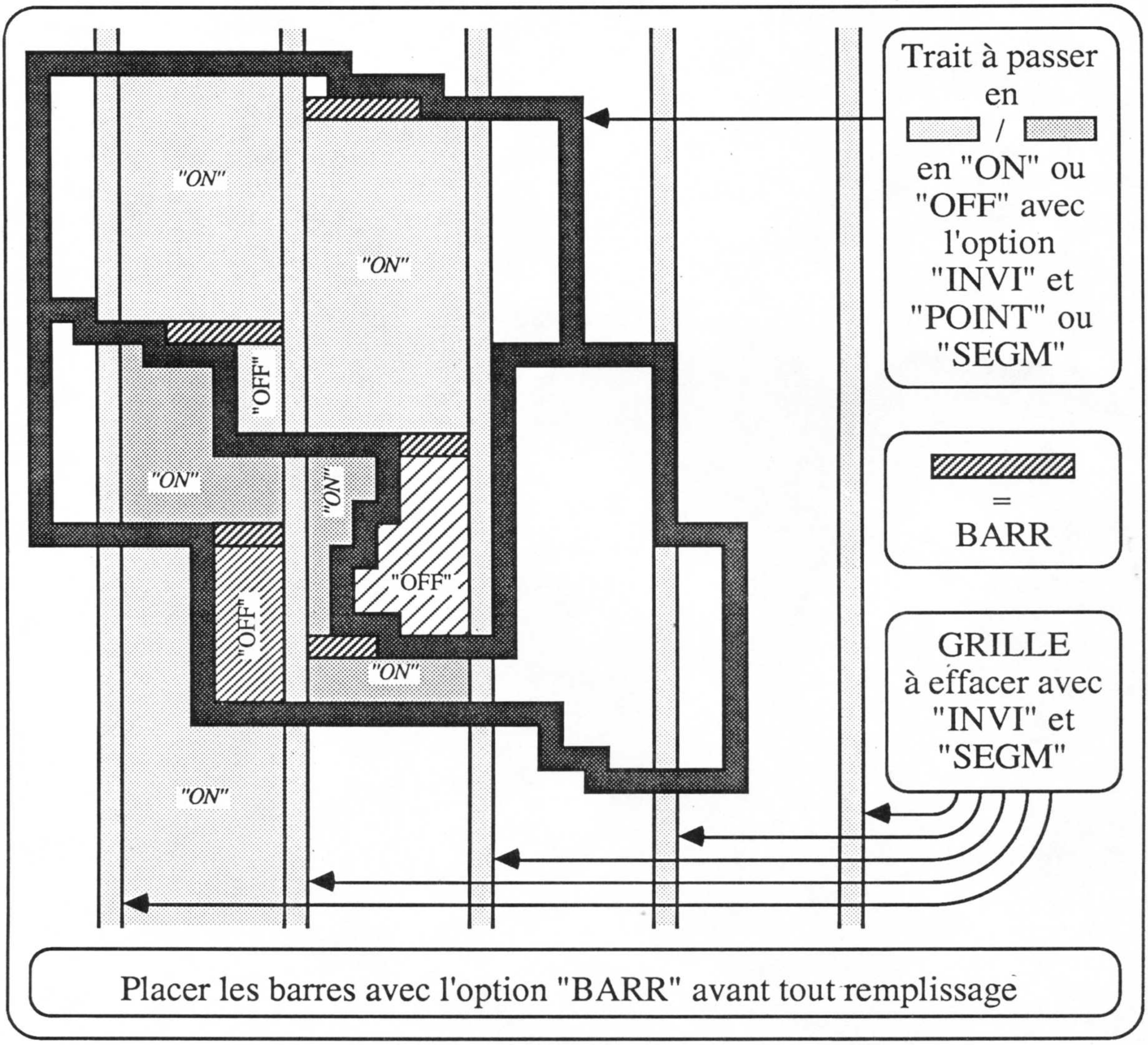
```
278 GOTO 232
279 IF T<>88 THEN 287
280 GOSUB 355
281 IF YA+DY>17 THEN 285
282 Y=YA+DY
283 X=XA
284 GOTO 232
285 Y=17
286 GOTO 232
287 IF T<>86 THEN 293
288 GOSUB 353
289 VE=1
290 GOTO 165
291 VE=0
292 GOTO 453
293 IF T<>73 THEN 235
294 AN=1
295 M$="partie annulee"
296 GOSUB 212
297 GOSUB 221
298 GOTO 469
299 CALL GCHAR(Y,X,A)
300 IF A<>48 THEN 304
301 A=152
302 G=52
303 GOTO 338
304 IF A<>56 THEN 308
305 A=48
306 G=60
307 GOTO 338
308 IF A<>64 THEN 312
309 A=56
310 G=68
311 GOTO 338
312 IF A<>72 THEN 316
313 A=64
314 G=76
315 GOTO 338
316 IF A<>80 THEN 320
317 A=72
318 G=84
319 GOTO 338
320 IF A<>88 THEN 324
321 A=80
322 G=92
323 GOTO 338
324 IF A<>128 THEN 328
325 A=88
326 G=132
327 GOTO 338
328 IF A<>136 THEN 332
329 A=128
330 G=140
331 GOTO 338
332 IF A<>144 THEN 336
333 A=136
334 G=148
335 GOTO 338
336 A=144
337 G=156
338 IF P<=NP THEN 390
339 CALL SOUND(5,1940,18)
340 FOR I=X TO X+1
341 CALL HCHAR(Y,I,G)
342 CALL HCHAR(Y+1,I,G+2)
343 G=G+1
344 NEXT I
345 RETURN
346 FOR I=1 TO 5
347 CALL SCREEN(12)
348 CALL SOUND(10,110,5)
349 CALL SCREEN(10)
350 NEXT I
351 CALL SCREEN(13)
352 GOTO 459
353 YA=Y
354 XA=X
355 CALL GCHAR(YA,XA,A)
356 IF A<>52 THEN 359
357 C=48
358 GOTO 384
359 IF A<>60 THEN 362
360 C=56
361 GOTO 384
362 IF A<>68 THEN 365
363 C=64
364 GOTO 384
365 IF A<>76 THEN 368
366 C=72
367 GOTO 384
368 IF A<>84 THEN 371
369 C=80
370 GOTO 384
371 IF A<>92 THEN 374
372 C=88
373 GOTO 384
374 IF A<>132 THEN 377
375 C=128
376 GOTO 384
377 IF A<>140 THEN 380
378 C=136
379 GOTO 384
380 IF A<>148 THEN 383
381 C=144
382 GOTO 384
383 C=152
384 FOR I=XA TO XA+1
385 CALL HCHAR(YA,I,C)
386 CALL HCHAR(YA+1,I,C+2)
387 C=C+1
388 NEXT I
389 RETURN
390 CALL SOUND(130,-4,5,40000,5,40
   000,5,110,5)
391 FOR I=X TO X+1
392 CALL HCHAR(Y,I,A)
393 CALL HCHAR(Y+1,I,A+2)
394 A=A+1
395 NEXT I
396 F=2500
```

```
397 IF Y=7 THEN 401
398 U=Y-2
399 V=X
400 GOSUB 416
401 IF X=21 THEN 405
402 U=Y
403 V=X+2
404 GOSUB 416
405 IF Y=17 THEN 409
406 U=Y+2
407 V=X
408 GOSUB 416
409 IF X=11 THEN 413
410 U=Y
411 V=X-2
412 GOSUB 416
413 IF P=NP THEN 206
414 IF P<NP THEN 205
415 GOTO 225
416 CALL GCHAR(U,V,A)
417 IF A>48 THEN 420
418 A=56
419 GOTO 445
420 IF A>56 THEN 423
421 A=64
422 GOTO 445
423 IF A>64 THEN 426
424 A=72
425 GOTO 445
426 IF A>72 THEN 429
427 A=80
428 GOTO 445
429 IF A>80 THEN 432
430 A=88
431 GOTO 445
432 IF A>88 THEN 435
433 A=128
434 GOTO 445
435 IF A>128 THEN 438
436 A=136
437 GOTO 445
438 IF A>136 THEN 441
439 A=144
440 GOTO 445
441 IF A>144 THEN 444
442 A=152
443 GOTO 445
444 A=48
445 CALL SOUND(100,F,15)
446 FOR I=V TO V+1
447 CALL HCHAR(U,I,A)
448 CALL HCHAR(U+1,I,A+2)
449 A=A+1
450 NEXT I
451 F=F+200
452 RETURN
453 PG=1
454 M$="bravo c est reussi"
455 GOSUB 212
456 GOSUB 52
457 GOSUB 221
458 GOTO 469
459 M$="permutations incompletes"
460 GOSUB 212
461 GOSUB 221
462 M$="continuez la partie"
463 GOSUB 212
464 GOSUB 221
465 FOR Y=7 TO 19 STEP 2
466 CALL HCHAR(Y,25,32)
467 NEXT Y
468 GOTO 223
469 M$="nouvelle partie   o n"
470 GOSUB 212
471 CALL KEY(3,T,S)
472 IF S=0 THEN 471
473 IF T<>78 THEN 491
474 CALL CLEAR
475 IF AN=1 THEN 477
476 IF PG=1 THEN 480
477 PRINT "la prochaine fois essay
   ez de"::"      terminer la parti
   e"::::"         bonne  journee"::
   :::
478 GOSUB 52
479 GOTO 483
480 PRINT " vous etes un partenair
   e de"::"         premiere force":
   :::"je suis impatient de rejoue
   r"::
481 PRINT :::"           a tres biento
   t"
482 GOSUB 52
483 CALL CLEAR
484 NCJ=NP*10
485 PRINT "  vous avez joue";CJ;"f
ois"
486 PRINT ::::"   vous deviez reus
   sir en"::TAB(9);NCJ;" coups":::
   :
487 END
488 CALL CLEAR
489 PRINT "lors de la prochaine pa
   rtie"::"  essayer de mieux etud
   ier"::"  le jeu avant chaque co
   up"::::
490 GOTO 481
491 IF T<>79 THEN 471
492 GOSUB 221
493 CJ=0
494 PG=0
495 AN=0
496 FOR X=11 TO 22
497 CALL VCHAR(7,X,32,12)
498 NEXT X
499 CALL HCHAR(1,3,32,12)
500 FOR X=16 TO 17
501 CALL VCHAR(1,X,32,2)
502 NEXT X
503 GOTO 122
```

Quatre nouvelles routines pour le crayon optique

Bernard Bezuel

Avant de vous présenter les quatre routines à ajouter à celles que nous avons publiées dans le numéro 8 de "99 Magazine", il convient de combler une lacune de ce même numéro. En effet, afin que l'image en cours ne soit pas altérée par les entrées/sorties (lors de l'utilisation des options "ENRE" et "RECH"), il faut utiliser CALL FILES (2). On peut donc imaginer un programme de chargement qui aurait pour forme :



```
100  CALL INIT
110  CALL FILES(2)
120  CALL LOAD("DSK1.CR
     AYON/O")
130  END
```

Routine "GRIL"

La principale utilité de cette routine est le remplissage de surfaces avec deux couleurs différentes sur un même octet (une ligne de huit pixels). Chaque ligne verticale tracée à l'écran représente le bit de poids fort d'un octet (correspondant au pixel le plus à gauche d'une ligne de huit pixels). En jouant sur les options "FOND" et "REMP", il est possible de colorier toute une surface sans altérer la couleur d'une surface voisine, dans le même octet.

Routine "BARR"

Sur une même bande verticale de huit pixels, il est également possible de jouer avec les options "FOND" et "REMP", en fragmentant avec l'option "BARR".

Routine "CERC"

Cette routine autorise le tracé de cercles à l'écran. Elle est très simple d'emploi, puisqu'il suffit de lui donner le centre, suivi du rayon. Cette routine n'utilisant pas les fonctions trigonométriques du TI, un algorithme a été développé pour l'occasion.

Routine "CADR"

Il s'agit d'une routine de tracé de rectangle, pour laquelle il convient d'indiquer les positions de deux angles opposées.

Mise à jour du programme

Vous trouverez, joint à cet article, une liste des modifications à apporter aux sources des routines publiées dans le précédent numéro, et ce afin de permettre l'implantation des quatre nouvelles fonctions.

99

Source GRIL/S

```
**  GRILLE SUR ECRAN **         GRIHG .......
GRIHG   DATA 0                  Y1,X1 .......
GRIBD   DATA 0                        .......
GRILLE  MOV  @DP10,@DP10              ....... Y2,X2
        JNE  GRI05                    ....... GRIBD
        MOV  @COORD,@GRIHG SAUVEGARDE SOMMET
*                          GRILLE
        INC  @DP10
        BLWP @LIGNE        AFFICHAGE SOMMET
        CLR  @DP2          DRAPEAU SEGMENT
        JMP  GRI25
GRI05   MOV  @COORD,@GRIBD SAUVEGARDE SOMMET
*                          GRILLE
        CLR  @DP10
        MOV  @OUTECR,@ATRIBU+36
        LI   R0,TABLUT+36
        LI   R1,ATRIBU+36
        LI   R2,4
        BLWP @VMBW
        CLR  R0
        CLR  R1
        CLR  R2
        CLR  R3
        MOVB @GRIHG,R1     R1=Y1
        MOVB @GRIHG+1,R0   R0=X1
        MOVB @GRIBD,R3     R3=Y2
        MOVB @GRIBD+1,R2   R2=X2
        SWPB R0
        SWPB R1
        SWPB R2
        SWPB R3
        ANDI R0,>00F8      8*INT(R0/8)
        MOV  R0,R4         SAUVEGARDE X1
        C    R0,R2
        JLT  GRI10
        LI   R6,-8
        JMP  GRI11
GRI10   LI   R6,8
GRI11   C    R1,R3
        JLT  GRI12
        LI   R7,-1
        JMP  GRI15
GRI12   LI   R7,1
```

```
GRI15  MOV   R3,R5
       S     R1,R5
       ABS   R5
       CI    R5,1
       JLT   GRI25
       A     R7,R1
       MOV   R4,R0
GRI20  MOV   R2,R5
       S     R0,R5
       ABS   R5
       CI    R5,8
       JLT   GRI15
       A     R6,R0
GRI21  MOV   R1,@SVY
       MOV   R0,@SVX
       BLWP  @POINTC
       JMP   GRI20
GRI25  B     *R11
```

Source BARR/S

DROITE HORIZONTALE ENTRE 2 POINTS ALLUMES

```
RTBARR DATA 0
BARRE  MOV   R11,@RTBARR
       CLR   R10
       MOVB  @COORD+1,R10
       SWPB  R10
       CLR   R1
       MOVB  @COORD,R1
       SWPB  R1
       BL    @REBIT
       JEQ   BA15
BA05   INC   R10
       CI    R10,252
       JGT   BA10
       BL    @REBIT
       JNE   BA05
BA10   DEC   R10
       MOV   R10,@DAV
       BL    @RE1LIN
BA15   MOV   @RTBARR,R11
       B     *R11
```

Source CERC/S

```
** CERC/S ** RAYON MAXIMUM >4000
RTCERC DATA 0
* SAISIE CENTRE
CERCLE MOV   R11,@RTCERC
       MOV   @DP2,@DP2
       JNE   CER05
       MOVB  @COORD,@CENTRY+1
       MOVB  @COORD+1,@CENTRX+1
       INC   @DP2
       MOVB  @YLUTIN+5,@ATRIBU+36
       MOVB  @ATRIBU+13,@ATRIBU+37
       LI    R0,TABLUT+36
       LI    R1,ATRIBU+36
       LI    R2,4
       BLWP  @VMBW
       B     *R11
* SAISIE DU RAYON
CER05  MOV   @OUTECR,@ATRIBU+36
       LI    R0,TABLUT+36
       LI    R1,ATRIBU+36
       LI    R2,4
```

```
       BLWP @VMBW
       CLR  @DP2
* CALCUL DE DELTAX AU CARRE
       CLR  R3
       MOVB @COORD+1,R3
       SWPB R3
       S    @CENTRX,R3
       ABS  R3
       SLA  R3,2
       MOV  R3,R4
       MPY  R3,R4        RESULTAT DANS R4/R5
* CALCUL DE DELTAY AU CARRE
       CLR  R6
       MOVB @COORD,R6
       SWPB R6
       S    @CENTRY,R6
       ABS  R6
       SLA  R6,2
       MOV  R6,R7
       LI   R0,20
       MPY  R0,R7
       LI   R0,29
       DIV  R0,R7
       MOV  R7,R0
       MPY  R0,R7        RESULTAT DANS R7/R8
       BL   @CER55
* RAYON=RACINE CARREE DE LA SOMME PRECEDENTE
       MOV  R0,R0
       JEQ  CER10
       MOV  @SQ1,@RAYON4 PREMIERE RACINE
*                        APPROCHEE
       JMP  CER15
```

```
CER10  MOV  R1,R12
       SRA  R12,1
       MOV  R12,@RAYON4  PREMIERE RACINE
*                        APPROCHEE
CER15  BL   @CER40
       MOV  R2,@RAYON4   RACINE APPROCHEE POUR
*                        PROCHAINE ITERATION
       LI   R0,29
       MPY  R0,R2
       LI   R0,20
       DIV  R0,R2
       SRA  R2,2
       MOV  R2,@RAYON
       JEQ  CER35
* TRACE DU CERCLE
       BL   @CER65
       BL   @CER70
       MOV  @RAYON,R6
       SLA  R6,2
       CLR  R3
CER20  AI   R3,4
       INC  @XPT1
       INC  @XPT2
       DEC  @XPT3
       DEC  @XPT4
CER25  MOV  R3,R4
       MPY  R3,R4
       MOV  R6,R7
       LI   R0,20
       MPY  R0,R7
       LI   R0,29
       DIV  R0,R7
```

```
       MOV  R7,R0
       MPY  R0,R7
       BL   @CER55
       BL   @CER40
       S    @RAYON4,R2
       CI   R2,1
       JLT  CER30
* RETOUR EN ARRIERE D'UN POINT
       DEC  @XPT1
       DEC  @XPT2
       INC  @XPT3
       INC  @XPT4
       INC  @YPT1
       INC  @YPT4
       DEC  @YPT2
       DEC  @YPT3
       AI   R6,-4
       AI   R3,-4
       JMP  CER25
* AFFICHAGE DU DERNIER POINT
CER30  BL   @CER70
       MOV  R6,R6
       JNE  CER20
* FIN DU CALCUL
CER35  MOV  @RTCERC,R11
       B    *R11
* RACINE CARRE DU DOUBLE-MOT R0,R1
* SQR(N)=(sqr+N/sqr)/2 ,sqr=RACINE APPROCHEE
* SORTIE RACINE DANS R2
CER40  LI   R7,30        NOMBRE D'ITERATION
*                        MAXIMUM
       LI   R8,2         DIVISEUR
       MOV  @RAYON4,R2   PREMIERE RACINE
*                        APPROCHEE
CER45  MOV  R0,R4
       MOV  R1,R5
       DIV  R2,R4
       A    R2,R4
       MOV  R4,R5
       CLR  R4
       DIV  R8,R4
       C    R4,R2
       JEQ  CER50
       MOV  R4,R2        NOUVELLE RACINE
*                        APPROCHEE
       JEQ  CER50
       DEC  R7
       JNE  CER45
CER50  B    *R11
* SOMME DES 2 CARRES
CER55  CLR  R0
       A    R5,R8
       JNC  CER60
       INC  R0
CER60  A    R4,R0
       A    R7,R0
       MOV  R8,R1
       B    *R11
* INITIALISATION DU TRACE
CER65  MOV  @CENTRX,@XPT1
       MOV  @CENTRX,@XPT2
       MOV  @CENTRX,@XPT3
       MOV  @CENTRX,@XPT4
       MOV  @CENTRY,@YPT1
```

```
       MOV   @CENTRY,@YPT2
       MOV   @CENTRY,@YPT3
       MOV   @CENTRY,@YPT4
       A     @RAYON,@YPT2
       A     @RAYON,@YPT3
       S     @RAYON,@YPT1
       S     @RAYON,@YPT4
       B     *R11
* AFFICHAGE DES 4 POINTS SYMETRIQUES
CER70  MOV   @XPT1,@SVX
       MOV   @YPT1,@SVY
       BLWP  @POINTC
       MOV   @XPT2,@SVX
       MOV   @YPT2,@SVY
       BLWP  @POINTC
       MOV   @XPT3,@SVX
       MOV   @YPT3,@SVY
       BLWP  @POINTC
       MOV   @XPT4,@SVX
       MOV   @YPT4,@SVY
       BLWP  @POINTC
       B     *R11
```

Source CADR/S

```
** CADR/S **                CAHG 3-----2
CAHG   DATA 0                    |     |
CABD   DATA 0                    |     |
CADRE  MOV   @DP9,@DP9           |     |
       JNE   CA05                4-----1 CABD
       INC   @DP9
       MOV   @COORD,@CAHG   SAUVEGERDE SOMMET 3
       BLWP  @LIGNE         SOMMET 3 DANS @LIGNES
*                           ET AFFICHAGE
       JMP   CA10
CA05   MOV   @COORD,@CABD   SOMMET 1 ET SAUVEGARDE
*                           COORD/MIRE
       MOVB  @CAHG,@COORD   SOMMET 2 DANS @COORD
       BLWP  @LIGNE         COTE 2-3
       MOV   @CABD,@LIGNES  SOMMET 1 DANS @LIGNES
       INC   @DP2
       BLWP  @LIGNE         COTE 1-2
       MOV   @CAHG,@COORD   SOMMET 3
       MOVB  @CABD,@COORD   SOMMET 4 DANS @COORD
       INC   @DP2
       BLWP  @LIGNE         COTE 1-4
       MOV   @CAHG,@LIGNES  SOMMET 3 DANS @LIGNES
       INC   @DP2
       BLWP  @LIGNE         COTE 3-4
       CLR   @DP9
       MOV   @CABD,@COORD   RESTITUTION COORD/MIRE
CA10   B     *R11
```

Liste des modifications à apporter aux sources publiés dans le numéro 8 de "99 Magazine"

```
NOUVELLES ROUTINES :CADRE,GRILLE,CERCLE,BARRE

#AL# ANCIENNE LIGNE EXISTANTE A CONSERVER
```

```
#LM# ANCIENNE LIGNE EXISTANTE A MODIFIER
#NL# NOUVELLE LIGNE A CREER
#LS# ANCIENNE LIGNE A SUPPRIMER

NOTA : POUR LA SUPRESSION DES LIGNES PLACER * AU
----   DEBUT DE LA LIGNE PUIS VERIFIER
       L'ENSEMBLE DES CORRECTIONS.
       EFFACER ENSUITE CES LIGNES.

********
*COPY/S*
********
0001         COPY "DSK1.REFE/S" #AL#
0002         COPY "DSK1.CADR/S" #NL#
0003         COPY "DSK1.GRIL/S" #NL#
0004         COPY "DSK1.CERC/S" #NL#
0005         COPY "DSK1.BARR/S" #NL#
0006         COPY "DSK1.REMP/S" #AL#

********
*REFE/S*
********
0032         REF  DSRLNK,KSCAN,VWTR #AL#
0033 *       AORG >2700 #LS# TAMPON ASCII (512
                        ADRESSES MEM)
0034 PAB     EQU  >1F00 #AL# BLOCK POUR ACCES
                        DISQUE
0159 DP1     DATA 0     #AL# DRAPEAU TRACE
                        CONTINU
0160 DP2     DATA 0     #LM# DRAPEAU TRACE DE
                        SEGMENT OU CERCLE
0161 DP3     DATA 1     #AL# DRAPEAU
                        EFFACEMENT
0166 DP8     DATA 0     #AL# DRAPEAU ECRITURE /
                        LECTURE SUR D
0167 DP9     DATA 0     #LM# DRAPEAU CADRE
0168 DP10    DATA 0     #NL# DRAPEAU GRILLE
0169 P1      DATA 1     #AL#
0170 M1      DATA -1    #AL#
0171 CENTRY DATA 0  CENTRE DU CERCLE       #NL#
0172 CENTRX DATA 0  CENTRE DU CERCLE       #NL#
0173 RAYON  DATA 0  RAYON DU CERCLE        #NL#
0174 RAYON4 DATA 0  4*RAYON                #NL#
0175 SQ1    DATA >2000  RACINE APPROCHEE   #NL#
0176 XPT1   DATA 0 *                       #NL#
0177 YPT1   DATA 0 *                       #NL#
0178 XPT2   DATA 0 *                       #NL#
0179 YPT2   DATA 0 * 4 POINTS SYMETRIQUES  #NL#
0180 XPT3   DATA 0 * DU CERCLE             #NL#
0181 YPT3   DATA 0 *                       #NL#
0182 XPT4   DATA 0 *                       #NL#
0183 YPT4   DATA 0 *                       #NL#

********
*REMP/S*
********
0047          BL   @REBIT   #AL#
0048          JEQ  RE105    #LM#
0049 RE110    BL   @RE1LIN  #AL#

********
*CRAY/S*
********
0179          JEQ  CR165     #AL#  DRAPEAU 0 TRACE
```

```
                                     POINT PAR POINT
0180        BLWP @POINT    #LM#  TRACE CONTINU
0181 *      LWPI USRWS     #LS#
0182        JMP  CR170     #AL#
0183 CR165  MOV  @DP4,@DP4 #AL#  DRAPEAU
                                 REMPLISSAGE
                                 CURVILIGNE
0184        JEQ  CR170     #AL#  REMPLISSAGE
                                 CURVILIGNE
0185        INC  @DP2      #AL#
0186        BLWP @LIGNE    #LM#

********
*MENU/S*
********
0003 MENU10 BYTE 32,67,69,82,67,49  CERC1   #LM#
0008        DATA MEN100,MEN110,MEN120,MEN130,
                 MEN140,MENU10              #LM#
0009        DATA SAUT2,SAUT2,SAUT2,SAUT2,
                 SAUT2,SAUT2                #LM#
0010 MEN100 BYTE 32,70,69,82,77,49   FERM1  #LM#
0016        DATA RCERCL,RESE01,RESE01,RESE01,
                 RESE01,SAUT2               #LM#
0017 MEN110 BYTE 32,86,73,83,73,49   VISI1  #AL#
0021        BYTE 32,71,82,73,76,53,
                 32,84              GRIL5 T #LM#
0022        DATA MEN110,MEN110,MENU10,MEN110,
                 MEN110,MENU10              #AL#
0023        DATA RVISIB,REFFAC,RCOULE,RFOND,
                 RGRILL,SAUT2               #LM#
0024 MEN120 BYTE 32,65,88,69,83,49   AXES1  #AL#
0035        BYTE 32,66,65,82,82,53,
                 32,84              BARR5 T #LM#
0036        DATA MEN121,MEN121,MEN121,MEN121,
                 MEN121,MEN120              #AL#
0037        DATA RHORIZ,RVERTI,RESE01,RESE01,
                 RBARRE,SAUT2               #LM#
0038 MEN130 BYTE 32,34,79,78,34,49  "ON"1   #AL#
0051        DATA RCADRE,RESE01,RECRAN,RSURFA,
                 SAUT2,SAUT2                #LM#
0052 MEN141 BYTE 32,79,70,83,69,49  OFSE1   #AL#
0155 RPOINT BLWP @POINT ROUTINE POINT       #LM#
0156 *      LWPI USRWS                      #LS#
0157        JMP  SAUT2                      #AL#
0158 RAXES  BLWP @HORIZ ROUTINE HORIZONTALE #LM#
0159        BLWP @VERTI ROUTINE VERTICALE   #LM#
0160        JMP  SAUT2                      #AL#
0161 RLIGNE BLWP @LIGNE  ROUTINE SEGMENT    #LM#
0244        BLWP @LIGNE                     #LM#
0245        JMP  ME75                       #AL#
0246 RHORIZ MOVB @COORD+1,R1                #AL#
0247        MOVB @P1,@COORD+1               #AL#
0248        BLWP @HORIZ                     #LM#

0251 RVERTI MOVB @COORD,R1                  #AL#
0252        MOVB @OUTECR,@COORD             #AL#
0253        BLWP @VERTI                     #LM#
0256 RSURFA MOV  @COLORS,@SVCOL             #AL#
0257        BL   @NEWCOL                    #AL#
0258        BL   @REMPLI                    #AL#
0259        MOV  @SVCOL,@COLORS             #AL#
0260        JMP  ME75                       #NL#
0261 RCADRE BL   @CADRE                     #NL#
0262        JMP  ME75                       #NL#
```

```
0263 RGRILL BL   @GRILLE                    #NL#
0264        JMP  ME75                       #NL#
0265 RCERCL BL   @CERCLE                    #NL#
0266        JMP  ME75                       #NL#
0267 RBARRE BL   @BARRE                     #NL#
0268        JMP  ME75                       #NL#
0269 ME75   NEG  @SVCLAV                    #AL#

********
*GRAF/S*
********
0004 POINT  DATA GRAFWS,RPT   #LM# APPEL AVEC Y/X
                                   DANS COORD
0005 *      LWPI GRAFWS       #LS#
0006 *      MOV  @RTGRAF,R11  #LS#
0007 RPT    CLR  @RTGRAF      #NL# APPEL
                                   EXTERIEUR
0008        MOVB @COORD+1,R0  #AL# X DANS R0
 0014 POINTC DATA GRAFWS,RPTC #LM#
0015 *      LWPI GRAFWS       #LS#
0016 *      MOV  @RTGRAF,R11  #LS#
0017 RPTC   CLR  @RTGRAF      #NL# APPEL
                                   EXTERIEUR
0018        MOV  @SVX,R0      #AL#
0029 *POINT2 MOV  R11,@RTGRAF #LS# APPEL AVEC
                                   Y/X DANS
                                   SVY/SVX
0030 *      LWPI GRAFWS       #LS#
0031 *      MOV  @RTGRAF,R11  #LS#
0032 *      MOV  @SVX,R0      #LS#
0033 *      MOV  @SVY,R1      #LS#
0034 POINT3 MOV  R1,R4        #AL# APPEL INTERNE
                                                A GRAF,Y/X
                                                DANS R1/R0
0095        MOVB R6,@>8C00          #AL#
0096        MOV  @RTGRAF,@RTGRAF    #NL#
0097        JEQ  GR20               #NL#
0098        B    *R11               #NL#
0099 GR20   RTWP                    #LM#
0100 *ROUTINE TRACE D'AXES          #AL#
0101 HORIZ  DATA GRAFWS,RHOR        #LM#
0102 RHOR   INC  @RTGRAF            #NL#
0103        MOVB @COORD,R9    Y     #AL#
0112        JMP  GR35               #AL#
0113 VERTI  DATA GRAFWS,RVER        #LM#
0114 RVER   INC  @RTGRAF            #NL#
0115        MOVB @COORD+1,R8 X      #AL#
0123        JNE  GR30               #AL#
0124 GR35   RTWP                    #LM#
0125 *      B    *R11               #LS#
0126 *ROUTINE TRACE DE SEGMENT      #AL#
0127 LIGNE  DATA GRAFWS,RLIG        #NL#
0128 RLIG   MOV  @DP2,@DP2          #LM#
0129        JNE  LIGNE2             #LM#
                               1:DEUXIEME POINT DU
                               SEGMENT
0130        MOV  @COORD,@LIGNES     #AL#
                               0:PREMIER POINT DU
                               SEGMENT
0137        BLWP @VMBW            #AL# AFFICHAGE
                                       MIRE FIXE
0138        RTWP                  #LM#
0139 *GR40   LWPI GRAFWS          #LS#
0140 LIGNEC DATA GRAFWS,LIGNE2    #NL# ENTREE
```

```
                                                DIRECTE
                                                SANS TEST
                                                DP2
0141 LIGNE2 MOV   @P1,@RTGRAF          #NL#
0142        MOV   @OUTECR,@ATRIBU+36   #AL#
0192        JMP   GR65                 #AL#
0193 GR80   RTWP                       #LM#
0194 *      B     *R11                 #LS#
0195 EFFT   LI    R0,>2000           #AL# ADRESSE
                                          TABLE
                                          DES
                                          COULEURS
```

Code machine à entrer en mémoire avec EASY BUG (option M)

```
A000   00 00 00 00 00 00 00 00 FF EF EF EF EF FF EF FF
A010   FF D7 D7 D7 FF FF FF FF FF FF D7 83 D7 83 D7 FF
A020   FF EF C3 AF C7 EB 87 FF FF 9F 9B F7 EF D9 B9 FF
A030   FF DF AF AF DF AB B7 CB FF FF FB FB F7 FF FF FF
A040   FF FF EF DF DF DF EF FF FF FF F7 FB FB FB F7 FF
A050   FF FF D7 EF 83 EF D7 FF FF FF EF EF 83 EF EF FF
A060   FF FF FF FF CF CF EF DF FF FF FF FF 87 FF FF FF
A070   FF FF FF FF FF CF CF FF FF FB F7 EF DF BF FF
A080   FF E7 DB DB DB DB E7 FF FF E7 D7 F7 F7 F7 E3 FF
A090   FF E7 DB FB F7 EF C3 FF FF E3 FB F7 FB DB E7 FF
A0A0   FF F7 E7 D7 D7 C3 F7 FF FF E3 EF E7 FB DB E7 FF
A0B0   FF E7 DF C7 DB DB E7 FF FF C3 FB F7 EF EF EF FF
A0C0   FF E7 DB E7 DB DB E7 FF FF E7 DB E3 FB DB E7 FF
A0D0   FF FF FF EF FF FF EF FF FF FF FF EF FF CF EF DF
A0E0   FF FF F7 EF DF EF F7 FF FF FF FF FF C3 FF C3 FF
A0F0   FF FF DF EF F7 EF DF FF FF CF B7 EF EF FF EF FF
A100   FF FF E3 DB D3 DF E3 FF FF FF C7 B7 B7 B7 CB FF
A110   9F DF C7 DB DB DB 87 FF FF FF C7 BB BF BB C7 FF
A120   F3 F7 C7 B7 B7 B7 C3 FF FF FF C7 BB 83 BF C7 FF
A130   E7 DB DF 8F DF DF DF FF FF E3 DB DB E3 FB DB E7
A140   9F DF D7 CB DB DB DB FF EF FF 8F EF EF EF 83 FF
A150   F7 FF E7 F7 F7 F7 B7 CF DF DF DB D7 CF D7 DB FF
A160   CF EF EF EF EF EF 83 FF FF FF 57 AB AB AB AB FF
A170   FF FF A7 DB DB DB DB FF FF FF C7 BB BB BB C7 FF
A180   FF FF 87 DB DB C7 DF 8F FF FF C3 B7 B7 C7 F7 E3
A190   FF FF A7 DB DF DF DF FF FF FF C7 BF C7 FB 87 FF
A1A0   DF DF 87 DF DF DB E7 FF FF FF B7 B7 B7 B7 CF FF
A1B0   FF FF BB BB D7 D7 EF FF FF FF 95 D5 D5 D5 EB FF
A1C0   FF FF BB D7 EF D7 BB FF FF FF DB DB DB E3 FB C7
A1D0   FF FF 83 B7 EF DB 83 FF FF FF CF DF DF DF CF FF
A1E0   FF FF FF DF EF F7 FB FF FF FF E7 F7 F7 F7 E7 FF
A1F0   FF EF D7 BB FF FF FF FF FF FF FF FF FF FF C3 FF
A200   52 A0 31 20 1B A0 07 52 45 46 A0 02 44 53 52 4C
A210   4E 4B 2C 4B 53 43 41 4E 2C 56 57 54 52 A0 33 20
A220   00 00 00 00 00 00 00 00 20 00 21 00 31 00 41 00
A230   51 00 61 00 71 00 81 00 91 00 A1 00 B1 00 C1 00
A240   D1 00 E1 00 F1 00 01 00 11 00 AE 24 A2 4E 20 20
A250   20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
A260   20 20 20 20 20 20 20 20 20 20 20 20 20 20 08 38
A270   68 98 C8 F8 B7 AE AA A5 A5 A5 AA AE AE AE C0 00
A280   C0 00 22 20 21 01 11 10 12 02 00 00 00 00 AE 1B
A290   00 00 C0 1C 00 0F C0 1C 01 0F C0 1C 02 0F C0 18
A2A0   03 0F C0 13 04 0F C0 13 05 0F C0 13 06 0F C0 18
A2B0   07 0F C0 18 0A 01 C0 00 08 0F C0 00 08 08 C0 00
A2C0   09 0F C0 F8 0B 0F C0 F8 0B 0F C0 F8 0B 0F C0 F8
A2D0   0B 0F D0 00 20 30 7C 7F 7F 3F 3F 1F 03 0F 3F 3F
A2E0   1F 0F 03 00 1F 3F 3F 7F 7F FC 30 00 FE FC 7C 78
A2F0   38 30 00 00 F8 FC FC FE FE 3E 0C 04 C0 F0 F8 FC
A300   FC F0 C0 00 00 0C 3F FE FE FC FC F8 00 00 18 38
A310   3C 7C 7E FE 10 10 00 C6 00 10 10 00 00 00 38 38
A320   38 00 00 00 EE FE FE 7C FE FE EE 00 FF FF FF FF
A330   FF FF FF FF 00 00 00 00 00 00 00 01 00 00 00 00
```

```
A340  00 00 00 00 00 00 00 00 00 00 00 01 FF FF 00 00
A350  00 00 00 00 00 00 20 00 00 00 00 00 00 00 00 00
A360  00 00 00 00 00 00 00 00 00 00 00 00 C8 20 A3 46
A370  A3 46 16 08 05 A0 A3 46 C8 20 A2 8E A3 68 04 20
A380  B6 52 10 25 C8 20 A2 8E A3 6A D8 20 A3 68 A2 8E
A390  04 20 B6 52 C8 20 A3 6A A2 8A 05 A0 A3 38 04 20
A3A0  B6 52 C8 20 A3 68 A2 8E D8 20 A3 6A A2 8E 05 A0
A3B0  A3 38 04 20 B6 52 C8 20 A3 68 A2 8A 05 A0 A3 38
A3C0  04 20 B6 52 04 E0 A3 46 C8 20 A3 6A A2 8E 04 5B
A3D0  00 00 00 00 C8 20 A3 48 A3 48 16 0A C8 20 A2 8E
A3E0  A3 D0 05 A0 A3 48 04 20 B6 52 04 E0 A3 38 10 47
A3F0  C8 20 A2 8E A3 D2 04 E0 A3 48 C8 20 A2 80 A2 B6
A400  02 00 1B A4 02 01 A2 B6 02 02 00 04 04 20 21 10
A410  04 C0 04 C1 04 C2 04 C3 D0 60 A3 D0 D0 20 A3 D1
A420  D0 E0 A3 D2 D0 A0 A3 D3 06 C0 06 C1 06 C2 06 C3
A430  02 40 00 F8 C1 00 80 80 11 03 02 06 FF F8 10 02
A440  02 06 00 08 80 C1 11 03 02 07 FF FF 10 02 02 07
A450  00 01 C1 43 61 41 07 45 02 85 00 01 11 10 A0 47
A460  C0 04 C1 42 61 40 07 45 02 85 00 08 11 F2 A0 06
A470  C8 01 A2 22 C8 00 A2 24 04 20 B5 18 10 F2 04 5B
A480  00 00 C8 0B A4 80 C8 20 A3 38 A3 38 16 17 D8 20
A490  A2 8E A3 4F D8 20 A2 8F A3 51 05 A0 A3 38 D8 20
A4A0  A2 7A A2 B6 D8 20 A2 9F A2 B7 02 00 1B A4 02 01
A4B0  A2 B6 02 02 00 04 04 20 21 10 04 5B C8 20 A2 80
A4C0  A2 B6 02 00 1B A4 02 01 A2 B6 02 02 00 04 04 20
A4D0  21 10 04 E0 A3 38 04 C3 D0 E0 A2 8F 06 C3 60 E0
A4E0  A3 50 07 43 0A 23 C1 03 39 03 04 C6 D1 A0 A2 8E
A4F0  06 C6 61 A0 A3 4E 07 46 0A 26 C1 C6 02 00 00 14
A500  39 C0 02 00 00 1D 3D C0 C0 07 39 C0 06 A0 A5 EC
A510  C0 00 13 04 C8 20 A3 56 A3 54 10 04 C3 01 08 1C
A520  C8 0C A3 54 06 A0 A5 C4 C8 02 A3 54 02 00 00 1D
A530  38 80 02 00 00 14 3C 80 08 22 C8 02 A3 52 13 3F
A540  06 A0 A5 FC 06 A0 A6 46 C1 A0 A3 52 0A 26 04 C3
A550  02 23 00 04 05 A0 A3 58 05 A0 A3 5C 06 20 A3 60
A560  06 20 A3 64 C1 03 39 03 C1 C6 02 00 00 14 39 C0
A570  02 00 00 1D 3D C0 C0 07 39 C0 06 A0 A5 EC 06 A0
A580  A5 C4 60 A0 A3 54 02 82 00 01 11 15 06 20 A3 58
A590  06 20 A3 5C 05 A0 A3 60 05 A0 A3 64 05 A0 A3 5A
A5A0  05 A0 A3 66 06 20 A3 5E 06 20 A3 62 02 26 FF FC
A5B0  02 23 FF FC 10 D7 06 A0 A6 46 C1 86 16 C9 C2 E0
A5C0  A4 80 04 5B 02 07 00 1E 02 08 00 02 C0 A0 A3 54
A5D0  C1 00 C1 41 3D 02 A1 02 C1 44 04 C4 3D 08 80 84
A5E0  13 04 C0 84 13 02 06 07 16 F3 04 5B 04 C0 A2 05
A5F0  17 01 05 80 A0 04 A0 07 C0 48 04 5B C8 20 A3 50
A600  A3 58 C8 20 A3 50 A3 5C C8 20 A3 50 A3 60 C8 20
A610  A3 50 A3 64 C8 20 A3 4E A3 5A C8 20 A3 4E A3 5E
A620  C8 20 A3 4E A3 62 C8 20 A3 4E A3 66 A8 20 A3 52
A630  A3 5E A8 20 A3 52 A3 62 68 20 A3 52 A3 5A 68 20
A640  A3 52 A3 66 04 5B C8 20 A3 58 A2 24 C8 20 A3 5A
A650  A2 22 04 20 B5 18 C8 20 A3 5C A2 24 C8 20 A3 5E
A660  A2 22 04 20 B5 18 C8 20 A3 60 A2 24 C8 20 A3 62
A670  A2 22 04 20 B5 18 C8 20 A3 64 A2 24 C8 20 A3 66
A680  A2 22 04 20 B5 18 04 5B 00 00 C8 0B A6 88 04 CA
A690  D2 A0 A2 8F 06 CA 04 C1 D0 60 A2 8E 06 C1 06 A0
A6A0  A8 96 13 0C 05 8A 02 8A 00 FC 15 03 06 A0 A8 96
A6B0  16 F9 06 0A C8 0A A6 D0 06 A0 A8 34 C2 E0 A6 88
A6C0  04 5B 00 00 00 00 00 00 00 00 00 00 00 00 00 00
A6D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
A6E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
A6F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
A700  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
A710  00 00 00 00 00 00 C8 0B A6 C2 02 E0 A2 00 C8 20
A720  A2 8E A6 D6 B8 20 A3 4B A6 D6 C8 20 A3 4A A6 D8
A730  02 03 00 04 02 23 FF FC 11 78 04 C0 D0 23 A6 D7
A740  06 C0 04 C1 D0 63 A6 D6 06 C1 C8 23 A6 D8 A6 CC
A750  C8 00 A6 D2 C8 00 A6 D0 C2 80 06 A0 A8 96 13 EA
A760  06 A0 A8 34 02 81 00 04 11 E5 02 81 00 B4 15 E2
A770  C8 20 A3 4C A6 CE 88 20 A6 D2 A6 D4 13 1B 88 20
A780  A6 D2 A6 D4 15 08 C2 A0 A6 D4 C3 20 A6 D2 06 0C
A790  06 A0 A9 26 10 0F C2 A0 A6 D2 C3 20 A6 D4 05 20
A7A0  A6 CC A0 60 A6 CC 06 0C 06 A0 A9 26 05 20 A6 CC
A7B0  A0 60 A6 CC C8 20 A6 D4 A6 D2 A0 60 A6 CC C2 A0
```

```
A7C0  A6 D0 06 A0 A8 96 13 1F C8 20 A3 4A A6 CE 02 0C
A7D0  00 FD 05 20 A6 CC A2 A0 A6 CE 83 0A 13 03 06 A0
A7E0  A8 96 16 F9 C3 20 A6 D0 06 0A C8 0A A6 D0 05 8A
A7F0  A0 60 A6 CC 05 20 A6 CE 06 A0 A9 26 05 20 A6 CC
A800  A0 60 A6 CC 10 AD C8 20 A3 4C A6 CE 02 0C 00 0D
A810  A2 A0 A6 CE 83 0A 13 03 06 A0 A8 96 13 F9 88 0A
A820  A6 D2 11 88 C8 0A A6 D0 10 9B 02 E0 20 BA C2 E0
A830  A6 C2 04 5B C8 0B A6 C6 C0 20 A6 D0 C0 80 C2 80
A840  06 A0 A8 96 06 A0 A8 C8 02 84 01 00 16 1E 02 27
A850  FF F8 06 A0 A9 64 02 82 00 10 11 13 C2 49 16 11
A860  02 0A FF 00 06 A0 A9 7A D2 A0 A2 28 02 27 20 00
A870  06 A0 A9 7A 02 27 E0 00 02 27 FF F8 02 22 FF F8
A880  10 E8 02 04 00 01 06 A0 A8 C8 05 82 C8 02 A6 D4
A890  C2 E0 A6 C6 04 5B C8 0B A6 CA C0 0A C1 C1 0A 57
A8A0  E1 C1 02 47 FF 07 C1 00 02 44 00 07 A1 C0 61 C4
A8B0  06 A0 A9 64 02 04 00 80 02 40 00 07 13 01 08 04
A8C0  C2 E0 A6 CA 41 09 04 5B 41 09 13 0C E2 44 06 02
A8D0  02 82 00 0C 15 02 05 82 10 05 0A 14 02 84 01 00
A8E0  13 01 10 F2 C8 0B A6 C4 C2 89 06 CA 06 A0 A9 7A
A8F0  10 02 C8 0B A6 C4 02 27 20 00 06 C7 D8 07 8C 02
A900  06 C7 D8 07 8C 02 D2 A0 A2 28 D3 60 88 00 02 4A
A910  F0 00 02 4D 0F 00 F2 8D 06 A0 A9 7A 02 27 E0 00
A920  C2 E0 A6 C4 04 5B C8 0B A6 C8 A2 A0 A6 CE 83 0A
A930  13 16 06 A0 A8 96 13 F9 C8 CA A6 D6 06 C1 D8 C1
A940  A6 D6 06 C1 C8 E0 A6 CC A6 D8 02 23 00 04 A2 A0
A950  A6 CE 83 0A 13 04 06 A0 A8 96 16 F9 10 E6 C2 E0
A960  A6 C8 04 5B 06 C7 D8 07 8C 02 06 C7 D8 07 8C 02
A970  04 C9 D2 60 88 00 06 C9 04 5B C2 07 02 0C 40 00
A980  06 C8 D8 08 8C 02 06 C8 E2 0C D8 08 8C 02 10 00
A990  D8 0A 8C 00 04 5B 00 00 00 00 C8 0B A9 96 C1 20
A9A0  A2 20 02 84 12 00 13 60 04 C9 D8 20 A2 7A A2 BA
A9B0  D8 20 A2 9F A2 BB 02 00 1B A8 02 01 A2 BA 02 02
A9C0  00 04 04 20 21 10 02 08 07 D0 02 07 02 58 04 20
A9D0  21 08 D8 20 83 7C 83 7C 16 15 D1 20 83 75 02 84
A9E0  FF 00 13 0C 02 89 00 03 11 05 02 07 00 C8 06 07
A9F0  16 FE 10 09 06 07 16 EB 05 89 10 05 06 08 16 E7

AA00  04 60 AB 28 04 C9 D1 20 83 75 16 03 02 03 00 07
AA10  10 28 02 84 0E 00 16 02 04 C3 10 23 02 84 03 00
AA20  16 03 02 03 00 01 10 1D 02 84 06 00 16 03 02 03
AA30  00 02 10 17 02 84 05 00 16 03 02 03 00 03 10 11
AA40  02 84 04 00 16 03 02 03 00 04 10 0B 02 84 02 00
AA50  16 03 02 03 00 05 10 05 02 84 0F 00 16 04 02 03
AA60  00 06 06 A0 AB 2E 10 A1 02 02 00 04 02 0A 00 3F
AA70  04 CC 02 0D 00 04 D8 20 A2 76 A2 B2 02 00 1B 80
AA80  02 01 A2 92 04 C3 C1 C0 C2 01 02 09 00 14 02 00
AA90  1B A0 02 01 A2 B2 04 20 21 10 C0 07 C0 48 D4 63
AAA0  A2 75 04 20 21 10 04 C4 04 20 21 08 D1 20 83 75
AAB0  02 84 12 00 13 23 06 09 16 F7 C0 07 C0 48 D4 60
AAC0  A2 80 04 20 21 10 06 0A 13 22 C3 0C 13 10 04 CC
AAD0  C0 C3 13 04 06 03 61 CD 62 0D 10 D7 02 00 1B 9C
AAE0  02 01 A2 AE 02 03 00 07 C1 C0 C2 01 10 CE 02 83
AAF0  00 07 13 C4 05 83 A1 CD A2 0D 10 C7 06 A0 AB 2E
AB00  02 0A 00 27 05 8C 02 09 00 32 04 60 AA 8E 02 00
AB10  80 00 06 00 16 FE D8 20 A2 80 A2 B2 02 00 1B A0
AB20  02 01 A2 B2 04 20 21 10 C2 E0 A9 96 04 5B C8 0B
AB30  A9 98 04 C0 D0 23 A2 82 C1 00 02 44 02 00 16 0A
AB40  02 06 A2 75 02 01 FF FF 02 04 03 00 02 05 00 01
AB50  06 A0 AB EA C1 00 02 44 01 00 16 0A 02 06 A2 75
AB60  02 01 00 01 02 04 B5 00 02 05 00 01 06 A0 AB EA
AB70  C1 00 02 44 20 00 16 0A 02 06 A2 8F 02 01 FF FF
AB80  02 04 0D 00 02 05 00 04 06 A0 AB EA C1 00 02 44
AB90  10 00 16 0A 02 06 A2 8F 02 01 00 01 02 04 FC 00
ABA0  02 05 00 04 06 A0 AB EA D8 20 A2 7D A2 8E D8 20
ABB0  A2 76 A2 B2 D8 20 A2 9F A2 B3 C8 00 A2 24 C8 01
ABC0  A2 22 C8 20 A3 36 A3 36 13 03 04 20 B5 02 10 0A
ABD0  C8 20 A3 3C A3 3C 13 06 05 A0 A3 38 04 20 B6 52
ABE0  04 E0 A3 38 C2 E0 A9 98 04 5B 91 16 13 07 02 04
ABF0  00 09 06 C1 B5 81 A1 85 06 04 16 FC 04 5B 00 00
AC00  01 30 01 48 01 18 01 40 01 48 01 28 01 90 00 00
AC10  01 48 01 68 01 08 01 38 01 28 00 00 01 20 01 98
AC20  01 58 00 80 00 00 00 00 00 00 00 01 AC 24 80 80
AC30  00 00 00 0F 44 53 4B 58 2E 58 58 58 58 58 58 58
```

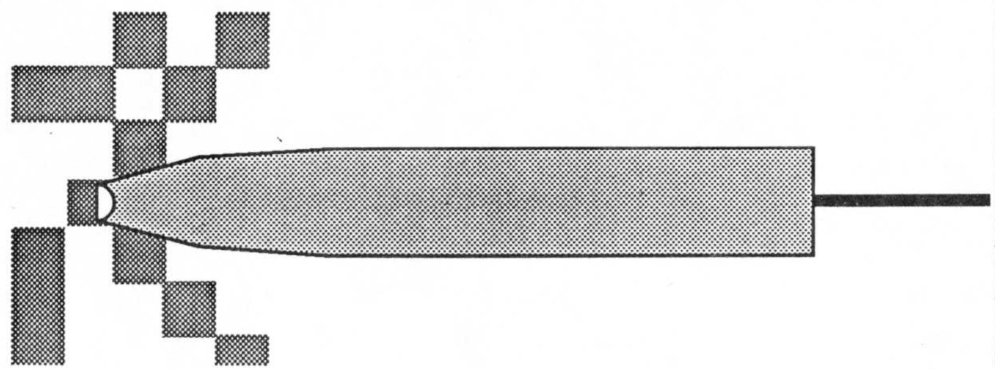
```
AC40   58 58 58 00 C8 20 A3 44 A3 44 13 05 02 00 03 00
AC50   C8 00 AB FE 10 04 02 00 02 00 C8 00 AB FE 04 C3
AC60   02 00 17 08 02 02 00 08 02 01 A0 00 A0 63 AC 00
AC70   04 20 21 10 02 20 00 08 05 C3 02 83 00 22 16 F4
AC80   02 00 00 00 D8 00 83 74 02 02 00 08 02 04 00 04
AC90   02 00 17 90 02 01 A1 F8 04 20 21 10 04 C3 04 20
ACA0   21 08 D0 E0 83 7C 13 FA D0 E0 83 75 02 84 00 04
ACB0   16 1C 02 83 31 00 16 05 02 01 A0 88 04 20 21 10
ACC0   10 07 02 83 32 00 16 EA 02 01 A0 90 04 20 21 10
ACD0   D8 03 AC 37 02 01 A0 70 02 20 00 08 04 20 21 10
ACE0   02 04 00 06 02 20 00 08 10 D5 02 83 08 00 16 0E
ACF0   02 84 00 06 13 C9 02 84 00 04 13 D0 02 01 A0 00
AD00   04 20 21 10 06 04 02 20 FF F8 10 C4 02 84 00 06
AD10   16 06 02 83 41 00 11 C2 02 83 5A 00 15 BF 02 83
AD20   20 00 13 BC 02 83 2E 00 13 B9 02 83 0D 00 13 12
AD30   D9 03 AC 33 02 01 A0 00 06 C3 02 23 FF E0 0A 33
AD40   A0 43 04 20 21 10 02 84 00 0F 13 A8 05 84 02 20
AD50   00 08 10 A0 06 04 06 C4 D8 04 AC 33 02 00 01 00
AD60   D8 00 83 74 02 00 1F 00 02 01 AC 26 02 02 00 20
AD70   04 20 21 10 02 06 1F 0D C8 06 83 56 04 20 21 20
AD80   00 08 D0 60 AB FE 02 00 1F 04 04 20 21 0C 02 00
AD90   1F 06 04 E0 AC 24 02 01 AC 24 02 02 00 02 04 20
ADA0   21 10 04 C3 C8 06 83 56 04 20 21 20 00 08 05 83
ADB0   A8 20 AC 22 AC 24 04 20 21 10 02 83 00 2E 16 F2
ADC0   02 01 20 00 C8 01 AC 24 02 01 AC 24 04 20 21 10
ADD0   C8 06 83 56 04 20 21 20 00 08 05 83 A8 20 AC 22
ADE0   AC 24 04 20 21 10 02 83 00 5C 16 F2 02 01 01 00
ADF0   02 00 1F 04 04 20 21 0C C8 06 83 56 04 20 21 20
AE00   00 08 02 03 00 1F 02 00 17 08 02 01 A0 00 02 02
AE10   00 08 04 20 21 10 02 20 00 08 06 03 16 FA 04 5B
AE20   AE 24 B7 68 20 43 45 52 43 31 20 54 52 41 49 32
AE30   20 52 45 43 54 33 20 43 55 52 56 34 20 55 54 49
AE40   4C 35 20 46 AE 5C AE 94 AE CC AF 3C AF 74 AE 24
AE50   B2 06 B2 06 B2 06 B2 06 B2 06 B2 06 20 46 45 52
AE60   4D 31 20 20 20 20 20 20 20 20 20 20 20 20 20 20
AE70   20 20 20 20 20 20 20 20 20 20 20 54 AE 5C AE 5C
AE80   AE 5C AE 5C AE 5C AE 24 B2 D8 B2 2A B2 2A B2 2A
AE90   B2 2A B2 06 20 56 49 53 49 31 20 45 46 46 41 32
AEA0   20 43 4F 55 4C 33 20 46 4F 4E 44 34 20 47 52 49
AEB0   4C 35 20 54 AE 94 AE 94 AE 24 AE 94 AE 94 AE 24
AEC0   B1 B2 B1 E2 B1 70 B1 76 B2 D2 B2 06 20 41 58 45
AED0   53 31 20 53 45 47 4D 32 20 44 52 4F 49 33 20 52
AEE0   45 4C 49 34 20 50 4F 49 4E 35 20 54 AE CC AE CC
AEF0   AF 04 AE CC AE CC AE 24 B1 4A B1 54 B2 06 B2 2A
AF00   B1 44 B2 06 20 48 4F 52 49 31 20 56 45 52 54 32
AF10   20 4F 52 49 45 33 20 44 45 4D 49 34 20 42 41 52
AF20   52 35 20 54 AF 04 AF 04 AF 04 AF 04 AF 04 AE CC
AF30   B2 8E B2 A2 B2 2A B2 2A B2 DE B2 06 20 22 4F 4E
AF40   22 31 20 22 4F 46 22 32 20 20 20 20 20 20 20 53
AF50   55 52 46 34 20 50 49 56 4F 35 20 54 AF 3C AF 3C
AF60   AF 3C AF 3C AF 3C AE 24 B2 2C B2 32 B2 2A B2 38
AF70   B2 70 B2 06 20 43 41 44 52 31 20 53 59 4D 42 32
AF80   20 45 43 52 41 33 20 52 45 4D 50 34 20 53 55 49
AF90   54 35 20 54 AF 74 AF 74 AE 24 AF 74 AF AC AE 24
AFA0   B2 CC B2 2A B2 26 B2 B6 B2 06 B2 06 20 4F 46 53
AFB0   45 31 20 5A 4F 4F 4D 32 20 47 52 4F 55 33 20 46
AFC0   45 4E 45 34 20 53 55 49 54 35 20 54 AF AC AF AC
AFD0   AF AC AF AC AF E4 AF 74 B2 2A B2 2A B2 2A B2 2A
AFE0   B2 06 B2 06 20 4F 56 4C 41 31 20 52 45 43 48 32
AFF0   20 45 4E 52 45 33 20 20 20 20 20 20 20 20 20 20
B000   20 20 20 54 AF E4 AE 24 AE 24 AF E4 AF E4 AF AC
B010   B2 2A B1 5A B1 5A B2 2A B2 2A B2 06 C8 0B AE 22
B020   C8 20 A2 4A AE 20 C8 20 A2 4C A2 4A 06 A0 B3 00
B030   C8 20 AE 20 A2 4A D1 20 A2 74 D8 04 A2 C2 D8 04
B040   A2 C6 D8 04 A2 CA D8 04 A2 CE 02 00 1B B0 02 01
B050   A2 C2 02 04 03 E8 06 04 16 FE 02 04 00 14 04 C5
B060   06 04 13 09 04 20 21 08 D1 60 83 75 02 85 12 00
B070   16 F7 04 60 B2 E8 04 C6 04 C7 D1 E6 A2 6E D8 07
B080   A2 C3 02 27 08 00 D8 07 A2 C7 02 27 08 00 D8 07
B090   A2 CB 02 27 08 00 D8 07 A2 CF 04 20 21 10 02 04
B0A0   00 18 04 20 21 08 D1 60 83 75 02 85 12 00 13 3C
B0B0   02 85 13 00 16 03 02 06 00 00 10 36 02 85 07 00
```

```
B0C0    16 03 02 06 00 01 10 30 02 85 08 00 16 03 02 06
B0D0    00 02 10 2A 02 85 09 00 16 03 02 06 00 03 10 24
B0E0    02 85 0A 00 16 03 02 06 00 04 10 1E 02 85 0B 00
B0F0    16 03 02 06 00 05 10 18 06 04 16 D3 05 86 02 86
B100    00 05 11 BB 13 04 06 A0 B3 36 04 60 B2 E8 02 07
B110    F8 00 D8 07 A2 C3 D8 07 A2 C7 D8 07 A2 CB D8 07
B120    A2 CF 04 20 21 10 10 BB 06 A0 B3 36 C1 20 AE 20
B130    02 24 00 20 0A 16 A1 06 C8 14 AE 20 02 24 00 0C
B140    C0 54 04 51 04 20 B5 02 10 5E 04 20 B6 0A 04 20
B150    B6 2E 10 59 04 20 B6 52 10 56 02 86 00 02 13 03
B160    05 A0 A3 44 10 02 04 E0 A3 44 06 A0 AC 44 10 4B
B170    06 A0 B3 60 10 48 C8 20 A3 40 A3 40 13 0C 04 E0
B180    A3 40 04 E0 A3 36 02 0C AE 94 C8 0C AE BA C8 20
B190    A2 26 A2 28 10 38 05 A0 A3 40 05 A0 A3 36 C8 20
B1A0    A2 28 A2 26 06 A0 B3 60 02 0C AE 24 C8 0C AE BA
B1B0    10 2A C8 20 A3 36 A3 36 16 0D 05 A0 A3 36 C8 20
B1C0    A3 3A A3 3A 16 18 02 0C AE 94 C8 0C AE B4 C8 0C
B1D0    AE B6 10 19 04 E0 A3 36 C8 20 A3 3A A3 3A 16 F3
B1E0    10 19 C8 20 A3 3A A3 3A 16 0F 05 A0 A3 3A C8 20
B1F0    A3 36 A3 36 13 E8 02 0C AE 24 C8 0C AE B4 02 0C
B200    AE 94 C8 0C AE B6 10 6E 04 E0 A3 3A C8 20 A3 36
B210    A3 36 16 D9 02 0C AE 94 C8 0C AE B4 02 0C AE 24
B220    C8 0C AE B6 10 5F 06 A0 B7 32 10 5C 04 E0 A3 40
B230    10 59 05 A0 A3 40 10 56 C8 20 A3 3E A3 3E 13 0E
B240    02 0C AF 3C C8 0C AF 62 04 E0 A3 40 04 E0 A3 3E
B250    04 E0 A3 3C C8 20 A2 26 A2 28 10 44 02 0C AE 24
B260    C8 0C AF 62 05 A0 A3 3E C8 20 A2 28 A2 26 10 3A
B270    C8 20 A3 3E A3 3E 13 36 05 A0 A3 3C 05 A0 A3 3A
B280    04 E0 A3 36 06 A0 B3 60 04 20 B6 52 10 2B D0 60
B290    A2 8F D8 20 A3 4A A2 8F 04 20 B6 0A D8 01 A2 8F
B2A0    10 21 D0 60 A2 8E D8 20 A2 80 A2 8E 04 20 B6 2E
B2B0    D8 01 A2 8E 10 17 C8 20 A2 28 A2 26 06 A0 B3 60
B2C0    06 A0 A7 16 C8 20 A2 26 A2 28 10 0C 06 A0 A3 6C
B2D0    10 09 06 A0 A3 D4 10 06 06 A0 A4 82 10 03 06 A0
B2E0    A6 8A 10 00 05 20 A2 20 C8 20 AE 20 A2 4A 06 A0
B2F0    B3 00 C2 E0 AE 22 02 01 80 00 06 01 16 FE 04 5B

B300    C1 60 A2 4A 02 04 00 20 02 00 17 00 02 02 00 08
B310    02 01 A0 00 04 C3 D0 D5 06 C3 02 23 FF E0 0A 33
B320    A0 43 04 20 21 10 05 85 02 20 00 08 06 04 16 F0
B330    02 02 00 10 04 5B 02 00 1B B0 02 01 A2 C2 02 02
B340    00 10 04 C4 D1 20 A2 80 D8 04 A2 C2 D8 04 A2 C6
B350    D8 04 A2 CA D8 04 A2 CE 04 20 21 10 04 5B 00 00
B360    C8 0B B3 5E C8 20 A2 4C A2 4A 06 A0 B3 00 04 C3
B370    02 00 37 08 C0 63 A2 2A 04 C4 04 20 21 0C 05 80
B380    05 84 02 84 00 06 16 F9 02 20 00 08 05 C0 05 C3
B390    02 83 00 1C 16 EF 02 00 17 08 02 02 00 08 02 01
B3A0    A3 2C 04 20 21 10 02 20 00 10 06 43 16 F8 02 00
B3B0    17 E8 02 01 A1 A0 02 02 00 08 04 20 21 10 02 00
B3C0    17 F8 02 01 A1 70 04 20 21 10 02 00 17 10 02 02
B3D0    00 08 02 03 00 1C 02 01 A1 00 04 20 21 10 02 20
B3E0    00 10 02 21 00 08 06 43 16 F8 02 04 80 00 06 04
B3F0    16 FE 04 C3 D8 03 83 74 04 20 21 08 D0 E0 83 75
B400    02 83 40 00 11 16 02 83 4E 00 11 0A 15 03 02 03
B410    00 0F 10 0A 02 83 54 00 16 0C 02 03 00 10 10 04
B420    02 04 40 00 60 C4 06 C3 02 04 01 00 D8 04 83 74
B430    10 49 02 04 01 00 D8 04 83 74 04 C4 04 20 21 08
B440    D1 20 83 75 02 84 12 00 16 D4 C8 20 A2 4C A2 4A
B450    06 A0 B3 00 02 04 20 00 06 04 16 FE 02 03 00 19
B460    04 20 21 08 D1 20 83 75 02 84 12 00 16 02 04 60
B470    B3 6E 06 03 16 F5 D8 20 A2 74 A2 CE 02 00 08 00
B480    D8 00 A2 CF 02 00 1B BC 02 01 A2 CE 04 20 21 10
B490    02 05 00 19 04 20 21 08 D1 20 83 75 02 84 12 00
B4A0    13 11 06 05 16 F7 02 05 10 00 B8 05 A2 CF 05 83
B4B0    02 83 00 10 16 EB D8 20 A2 80 A2 CE 04 20 21 10
B4C0    04 60 B3 6E D8 20 A2 80 A2 CE 04 20 21 10 05 84
B4D0    02 84 50 00 16 FC 0A 13 C8 23 A2 2A A2 28 02 00
B4E0    37 00 02 01 F1 00 04 20 21 0C 05 80 02 80 38 00
B4F0    16 FA C2 E0 B3 5E 04 5B 00 00 00 00 00 00 00 00
B500    00 00 A2 00 B5 06 04 E0 B4 F8 D0 20 A2 8F 09 80
B510    D0 60 A2 8E 09 81 10 15 A2 00 B5 1C 04 E0 B4 F8
B520    C0 20 A2 24 C0 60 A2 22 02 80 00 0D 11 6D 02 80
B530    00 FC 15 6A 02 81 00 03 11 67 02 81 00 B5 15 64
```

```
B540  10 00 C1 01 0A 54 E1 01 02 44 FF 07 C1 40 02 45
B550  00 07 A1 00 61 05 C8 20 A3 40 A3 40 13 15 02 05
B560  40 00 02 24 20 00 06 C4 D8 04 8C 02 06 C4 D8 04
B570  8C 02 D0 20 A2 28 D1 A0 88 00 02 46 F0 00 02 40
B580  F0 00 09 40 F1 80 10 31 06 C4 D8 04 8C 02 06 C4
B590  D8 04 8C 02 C0 05 D1 A0 88 00 02 03 80 00 0B 03
B5A0  02 05 40 00 C8 20 A3 3A A3 3A 16 02 51 83 10 1D
B5B0  F1 83 C0 04 02 20 20 00 06 C0 D8 00 8C 02 06 C0
B5C0  D8 00 8C 02 D0 60 A2 28 D0 E0 88 00 02 41 F0 00
B5D0  02 43 0F 00 F0 43 06 C0 D8 00 8C 02 06 C0 E0 05
B5E0  D8 00 8C 02 10 00 D8 01 8C 00 06 C4 D8 04 8C 02
B5F0  06 C4 E1 05 D8 04 8C 02 10 00 D8 06 8C 00 C8 20
B750  02 80 17 00 16 FA 04 5B 02 E0 20 BA 06 A0 B8 1E
B760  06 A0 B8 C4 06 A0 B0 1C 04 E0 A3 34 06 A0 B7 F0
B770  06 A0 B7 BE 03 00 00 02 03 00 00 00 05 A0 A3 34
B780  06 A0 B7 F0 02 01 00 64 06 01 13 EE 04 20 21 08
B790  04 C0 D0 20 83 75 02 80 FF 00 13 F6 C8 00 A2 20
B7A0  06 A0 B0 1C C8 20 A2 20 A2 20 11 E4 04 E0 A3 34
B7B0  06 A0 B7 F0 06 A0 B7 BE 06 A0 A9 9A 10 D5 D8 20
B7C0  A2 7A A2 BA D8 20 A2 9F A2 BB 02 00 1B A8 02 01
B7D0  A2 BA 02 02 00 04 04 20 21 10 02 03 4E 20 06 03
B7E0  10 00 16 FD C8 20 A2 80 A2 BA 04 20 21 10 04 5B
B7F0  C8 20 A3 34 A3 34 13 07 D8 20 A2 7A A2 BE D8 20
B800  A2 9F A2 BF 10 03 C8 20 A2 80 A2 BE 02 00 1B AC
B810  02 01 A2 BE 02 02 00 04 04 20 21 10 04 5B 02 00
B820  01 F0 04 20 21 1C 02 00 1C 00 04 C2 04 C1 04 C3
B830  04 20 21 0C 05 80 05 83 02 21 01 00 02 83 01 00
B840  16 F7 05 82 02 82 00 03 16 F1 04 C0 04 C1 04 20
B850  21 0C 05 80 02 80 18 00 16 FA 02 00 20 00 02 01
B860  11 00 04 20 21 0C 05 80 02 80 37 00 16 FA 02 00
B870  37 00 02 01 F1 00 04 20 21 0C 05 80 02 80 38 00
B880  16 FA 02 00 00 02 04 20 21 1C 02 00 01 E0 04 20
B890  21 1C 02 00 02 07 04 20 21 1C 02 00 03 FF 04 20
```

```
B600  B4 F8 B4 F8 13 01 04 5B 03 80 A2 00 B6 0E 05 A0
B610  B4 F8 D2 60 A2 8E 09 89 02 08 00 0D C0 08 C0 49
B620  06 A0 B5 42 05 88 02 88 00 FD 11 F8 10 11 A2 00
B630  B6 32 05 A0 B4 F8 D2 20 A2 8F 09 88 02 09 00 03
B640  C0 08 C0 49 06 A0 B5 42 05 89 02 89 00 B6 16 F8
B650  03 80 A2 00 B6 56 C8 20 A3 38 A3 38 16 17 C8 20
B660  A2 8E A2 8A C8 20 A3 4A A3 38 D8 20 A2 7A A2 B6
B670  D8 20 A2 9F A2 B7 02 00 1B A4 02 01 A2 B6 02 02
B680  00 04 04 20 21 10 03 80 A2 00 B6 8C C8 20 A3 4A
B690  B4 F8 C8 20 A2 80 A2 B6 02 00 1B A4 02 01 A2 B6
B6A0  02 02 00 04 04 20 21 10 C8 20 A2 8E A2 8C 04 E0
B6B0  A3 38 D2 60 A2 8B 09 89 D1 E0 A2 8D 09 87 C8 07
B6C0  B4 FE D2 A0 A2 8A 09 8A 05 8A D2 20 A2 8C 09 88
B6D0  05 88 C8 08 B5 00 04 CC C8 20 A3 4A B4 FA C8 20
B6E0  A3 4A B4 FC 61 C9 11 01 10 04 C8 20 A3 4C B4 FA
B6F0  05 07 C1 C7 16 01 07 0C 62 0A 11 01 10 04 C8 20
B700  A3 4C B4 FC 05 08 C0 09 C0 4A 06 01 06 A0 B5 42
B710  88 09 B4 FE 16 03 88 0A B5 00 13 0A C3 0C 11 04
B720  A2 60 B4 FA 63 08 10 EF A2 A0 B4 FC A3 07 10 EB
B730  03 80 02 00 20 00 02 01 11 00 04 20 21 0C 05 80
B740  02 80 37 00 16 FA 04 C0 04 C1 04 20 21 0C 05 80
B8A0  21 1C 02 00 04 03 04 20 21 1C 02 00 05 37 04 20
B8B0  21 1C 02 00 06 03 04 20 21 1C 02 00 07 01 04 20
B8C0  21 1C 04 5B 02 03 01 00 D8 03 83 74 02 00 18 00
B8D0  02 01 A2 D4 02 02 00 98 04 20 21 10 02 00 1B 80
B8E0  02 01 A2 92 02 02 00 80 04 20 21 10 04 5B
```



Tracé de courbes en haute résolution par splines cubiques

Gérard Santraille

Nous allons utiliser aujourd'hui l'ensemble des routines graphiques développées depuis le numéro quatre, pour créer et manipuler des "splines". Ne cherchez pas l'explication de ce terme dans le dictionnaire, elle n'y figure vraissemblablement pas. Les splines sont des familles de courbes permettant d'interpoler (ou d'extrapoler) par un tracé, un ensemble de points. Explicitons plus profondément cette définition un peu brutale.

Le problème est le suivant : disposant d'un jeu de données (résultats expérimentaux, cours du dollar, températures journalières) on aimerait dessiner à l'écran la meilleure représentation graphique de cet ensemble de données. Cette représentation graphique peut avoir toutes sortes d'utilités : interpolation entre deux valeurs, prévisions (extrapolation), etc…

Selon le traitement que l'on applique aux données, le tracé obtenu peut changer très sensiblement. En fait, il convient de définir correctement, pour une situation donnée, ce que l'on appelle : *la meilleure représentation graphique d'un jeu de données.*

Supposons que l'on veuille modéliser (c'est-à-dire paramétrer mathématiquement) le comportement d'un appareil donné. Dans le cas le plus élémentaire (voir figure 1) nous disposons d'un ensemble de couples de données représentant une série de valeurs de sortie (Y_i) de l'appareil pour une série de valeurs d'entrée (X_i).



Figure 1

Partant de cette famille de couples, on désire trouver la meilleure équation du type Y=f(X) permettant de retracer le comportement de l'appareil. Cette fonction unique pour l'ensemble de tous les points de coordonnées (X_i, Y_i) ne donnera probablement pas l'exacte valeur Y_i pour chaque X_i. Il ne s'agit que de la meilleure approximation. Le tracé simultané sur l'écran de cette fonction avec tous les points expérimentaux montrera que cette fonction ne "passe" pas par tous les points. Ce n'est d'ailleurs pas là le but recherché. L'intérêt de ce type de calcul (le traitement mathématique que l'on applique aux points pour évaluer la fonction s'appelle une régression) est de pouvoir substituer à un ensemble de points une équation algébrique beaucoup plus facile à manipuler (dérivation, intégra-

Programme de démonstration

```
100 REM *************
110 REM *           *
120 REM * Copyright *
130 REM *           *
140 REM *    99     *
150 REM * Magazine  *
160 REM *    &      *
170 REM *  Gerard   *
180 REM * Santraille *
190 REM *           *
200 REM *************
210 REM
220 REM
230 CALL CLEAR
240 OPTION BASE 1
250 DIM A(15,5),B(15,5
    ),NP(5),C(4,15),CP
    (5),CC(5),IB(5),IE
    (5),XR(6),LI(2,5)
260 REM   jusqu'a 15 p
    oints par courbe !
270 REM  jusqu'a 5 cou
    rbes differentes
280 REM   A     : absci
    sses
290 REM   B     : ordon
    nees
300 REM   NP    : nbre
    de points
310 REM   C     : coeff
    icients du polynom
    e d'interpolation
320 REM   C  peut egal
    ement contenir plu
    sieurs matrices de
     travail
330 REM   CP    : coule
    urs des points
340 REM   CC    : coule
    urs des courbes
350 REM  IB,IE : Condi
    tions limites
```

tion). L'unicité de la formulation mathématique pour tous les points implique que la courbe ainsi obtenue ne passe pas exactement pas les points qui ont servi à la calculer.

Il existe de très nombreuses méthodes permettant de calculer cette fonction après avoir choisi sa forme (le modèle). Nous en verrons sommairement plusieurs un peu plus loin.

Si,. au contraire, on désire privilégier la visualisation graphique d'une courbe calculée à partir d'un ensemble de points et non pas sa simplicité au niveau de la formulation mathématique, il est possible d'évaluer non pas une équation, mais une famille d'équations pour un jeu de données. Pour un ensemble de *n* points on peut ainsi définir *n-1* équations. Une équation donnée n'est alors valable que pour l'intervalle formé par deux points voisins. Ce type de régression, beaucoup plus lourd à manipuler algébriquement, permet d'effectuer un tracé passant *exactement* par tous les points. C'est une régression de ce type que nous verrons en détail avec l'analyse de la technique des "splines cubiques".

Le but de cet article n'est pas de comparer ces deux philosophies; il est cependant possible de donner quelques règles générales. Tous les scientifiques qui tentent de modéliser un processus physico-chimique utilisent généralement, pour des raisons élémentaires de mathématique, la première formulation (une seule équation pour tous les points). Il est beaucoup plus aisé de manipuler une équation qu'une famille. Bien évidemment, tout dépend de l'allure des données expérimentales. Si ces données révèlent une cassure importante dans leur allure, comme le montre la figure 2, il est préférable de découper ces données en deux ensembles et d'avoir deux équations relativement simples plutôt qu'une seule (généralement complexe) tentant de retracer correctement ces valeurs sur la totalité du domaine.

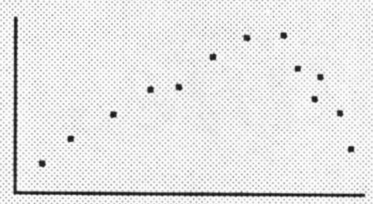


Figure 2

Une régression portant sur une seule ou sur peu d'équations permet par ailleurs de ne pas donner trop d'importance à un point isolé par rapport aux autres. Cette caractéristique est très importante lorsqu'il s'agit de mesures expérimentales sujettes à des erreurs de mesure.

Il n'existe pas de règle générale quant au choix du modèle (de la fonction). Pour des modélisations se rapportant à des phénomènes physiques connus, la théorie dicte généralement la forme de la fonction dont il faut calculer les divers paramètres. Souvent, la forme globale du nuage de points permet également de se faire une idée de la forme que peut avoir le modèle.
Ainsi, les diagrammes suivants (figures 3 et 4) suggèrent respectivement une fonction type logarithme et une fonction type sinus amorti.

```
360 REM   XR   : Matri
    ce de travail
370 REM   LI   : Valeu
    rs des derivees au
    x limites
380 CALL SCREEN(15)
390 REM  TEST derniere
     adresse libre mem
    oire basse
400 CALL PEEK(8264,S)
410 IF S=32 THEN 460
420 CALL INIT
430 CALL LOAD("DSK1.PL
    OT4/OBJ")
440 REM  couleur ecran
     par defaut
450 CALL LOAD(-24572,1
    )
460 GOSUB 810
470 CALL CLEAR
480 PRINT "Trace de co
    urbes par Splines"
490 PRINT " ou par lis
    sage polynomial"
500 PRINT "du troisiem
    e degre au sens"
510 PRINT "des moindre
    s carres"
520 PRINT "----------
    ---------------"
530 PRINT
540 PRINT
550 PRINT "1->Lire un
    jeu de donnees"
560 PRINT "2->Afficher
     les points"
570 PRINT "3->Trace (s
    plines)"
580 PRINT "4->Trace (m
    oindres carres)"
590 PRINT "5->Nouvelle
     image"
600 PRINT "6->Finir"
610 PRINT
620 PRINT
630 PRINT
640 PRINT
650 CALL SOUND(130,140
    0,0)
```

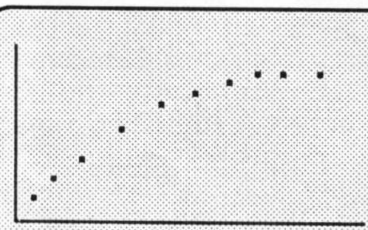


Figure 3

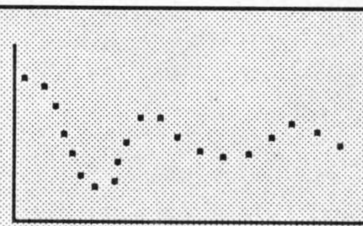


Figure 4

En l'absence de tout renseignement sur la forme du modèle à utiliser, il est habituel d'employer des polynômes. L'utilisation des polynômes pour des régressions est intéressante car ils sont aisément évaluables, dérivables et intégrables à l'aide des seules opérations arithmétiques de base. Un polynôme de degré *n* peut s'écrire selon :

$P_n(x)=$
$a_0 + a_1x + a_2x^2 + ... + a_nx^n$

La régression sur une telle fonction consiste, une fois que l'on a choisi l'ordre *n* du polynôme, à determiner la valeur des *n+1* coefficients a0, a1, a2,....., an.

Il convient d'être très prudent sur le choix du degré *n* du polynome. Pour lisser un jeu de *m* points, il peut être tentant d'utiliser un polynôme d'ordre *m-1* (on obtient alors un système de *m* équations à *m* inconnues que l'on résoud très classiquement, ou on utilise la technique du "polynôme de Lagrange"). Le résultat peut être catastrophique et donner une courbe de l'allure de la figure 5.



Figure 5

Polynômes de Lagrange

Considérons (Pi) un ensemble de *n* points distincts (le jeu de données sur lequel va porter la régression). Supposons que xi,yi soient les coordonnées de chaque point.

Le polynome
$L_i(x)=Pi\ (x-x_j)/(x_i-x_j)$
pour j=1,..,n et j≠i
(Pi signifie produit)
est un polynôme de degré *n-1* qui prend la valeur 0 pour tous les xj sauf pour xi où il prend la valeur 1.

Le polynôme
$P(x) = \sum y_iL_i$ pour i = 1, ..., n
est donc bien un polynôme d'ordre *n-1* (combinaison linéaire de polynômes d'ordre *n-1*) passant par tous les points Pi. En effet :
pour tout i on a bien $P(x_i)=y_i$

En fait, cette technique n'est pas employée, et l'on ne dépasse que très rarement un ordre supérieur à quatre. Examinons la méthode d'interpolation la plus répandue : la méthode des moindre carrés.

Méthode des moindres carrés

Cette technique qui s'applique à n'importe quelle autre forme de fonction que le polynôme conduit toujours à l'évaluation d'une seule équation pour l'ensemble des points.

L'idée est de calculer le polynôme d'ordre *n* le mieux ajusté à l'ensemble des points. Il faut choisir *n* très largement inférieur au nombre de points (habituellement n<4). Cet ajustement sera considéré optimal lorsque la somme des carrés des écarts entre les valeurs expérimentales et les valeurs calculées par le polynôme sera *minimale*.

Ainsi, si
$P_n(x)=a_0 + a_1x + a_2x^2 +... + a_nx^n$
est le polynôme d'ajustement d'ordre *n*, et *m* le nombre de points sur lesquels va porter la régression il convient de minimiser la fonction :

$$\sum_{i=0}^{m} [\sum_{j=0}^{n} a_jx_i^j - y_i]^2 = H$$

```
660 GOTO 680
670 CALL SOUND(130,220
    ,0)
680 CALL KEY(0,K,S)
690 IF S=0 THEN 680
700 IF (K<49)+(K>54)TH
    EN 670
710 ON K-48 GOSUB 840,
    1940,2730,2080,800
    ,3630
720 GOTO 470
730 FOR S=1 TO 4 STEP
    .5
740 CALL SOUND(40,300*
    S,0)
750 NEXT S
760 FOR S=1 TO 50
770 NEXT S
780 RETURN
790 REM  Initialisatio
    n image
800 PRINT "Effacement
    de l'image":"actue
    lle ..."
810 CALL LINK("EFFT")
820 RETURN
830 REM  Lecture Fichi
    er de donnees
840 CALL CLEAR
850 PRINT "Lecture d'u
    n jeu de donnees"
860 REM  valeurs par d
    efaut
870 ECR=1
880 CP(1)=9
890 CC(1)=15
900 IB(1)=0
910 IE(1)=0
920 PRINT
930 INPUT "Nom du fich
    ier : ":FILE$
940 IF FILE$<>"" THEN
    970
950 FILE$="DSK1.DATA"
960 GOTO 1140
970 IF SEG$(FILE$,1,3)
    <>"CAT" THEN 1120
980 I$=SEG$(FILE$,4,1)
990 IF (I$<>"1")*(I$<>
    "2")*(I$<>"3")THEN
     920
1000 PRINT ::"Liste de
     s fichiers DSK";I
     $:"--------------
     ---------"
1010 OPEN #2:"DSK"&I$&
     ".",INPUT ,RELATI
     VE,INTERNAL
1020 INPUT #2:A$,J,J,J
1030 PRINT "Nom du dis
     que : ";A$::
1040 INPUT #2:A$,T,J,J
```

H s'appelle la fonction objective.

Pour minimiser cette fonction on peut résoudre le système d'équations linéaires formé par les dérivées partielles de H par rapport aux a_j ou utiliser n'importe quelle autre technique classique d'optimisation. Il est bien évident que la régression sera d'autant plus aisée que l'ordre du polynôme sera faible et d'autant plus précise que le nombre de points sera élevé.
Il est par ailleurs possible de pondérer la somme des carrés par des coefficients dont les valeurs s'inspirent de deux considérations :

- régularité de l'espacement des points d'échantillonage;
- confiance en la précision du point.

La fonction objective devient alors :

$$\sum_{i=0}^{m} w_i \left[\sum_{j=0}^{n} a_j x_i^j - y_i \right]^2$$

On peut ainsi donner plus de "poids" a un point si on est sur de sa valeur ou s'il se trouve trop isolé afin de rectifier l'allure générale de la courbe.

La technique des moindres carrés dont nous venons de voir le principe est très usitée dès qu'il s'agit de calculer les paramètres d'un modèle dont la forme est bien connue. Pour une visualisation graphique d'un jeu de données dont on ne connait pas a priori l'expression du modèle, cette méthode peut conduire à des résultats très différents selon le choix du modèle ou plus simplement selon l'ordre du polynôme. Le programme Basic de démonstration permet de superposer à un jeu de données un polynôme de degré 3 dont les coefficients ont été calculés à l'aide de cette technique.

Nous allons voir maintenant plusieurs techniques donnant une représentation graphique passant par tous les points. Comme nous l'avons déjà vu, cette paramétrisation implique la définition de plusieurs équations différentes. Comme il est possible de faire passer un polynôme de degré *m-1* par un ensemble de *m* points, on peut envisager de découper cet ensemble de *m* points en *q* sous-ensembles de *p* points pour chacun desquels on calculera le polynôme de degré *p-1* passant exactement par tous les *p* points de ce sous-ensemble.

L'expression la plus élémentaire de cette technique consiste à joindre tous les points par des tronçons de droites. Il s'agit bien d'un découpage en série de deux points pour chacune desquelles on fait une regression par un polynôme d'ordre 1.

On dispose ainsi d'un ensemble d'équations linéaires permettant, comme le montre la figure 6, de retracer assez grossièrement l'allure de la courbe.

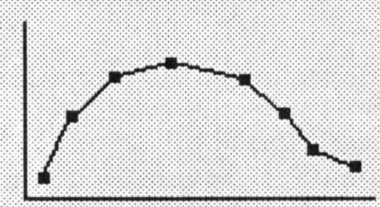


Figure 6

Pour tenter d'affiner notre représentation on peut décider non plus d'interpoler deux points voisins par un tronçon de droite, mais trois points voisins par un polynôme de degré 2 ou encore quatre points voisins par un polynome de degré 3. Ce type de représentation donne une certaine "rotondité" à la courbe mais comporte généralement d'assez fortes "cassures" aux points de jonction de deux équations algébriques différentes (voir figure 7).

Interpolation du troisième ordre pour les points médians.

```
1050 IF A$<>"" THEN 1090
1060 CLOSE #2
1070 PRINT :::
1080 GOTO 850
1090 IF ABS(T)-2 THEN 1040
1100 PRINT A$,
1110 GOTO 1040
1120 IF POS(FILE$,".",1)THEN 1140
1130 FILE$="DSK1."&FILE$
1140 IC=1
1150 P=0
1160 OPEN #1:FILE$,INPUT
1170 IF EOF(1)THEN 1790
1180 INPUT #1:DATA$
1190 IF ASC(DATA$)=59 THEN 1170
1200 TOTO$=SEG$(DATA$,1,3)
1210 IF TOTO$<>"IBE" THEN 1260
1220 IB(IC)=VAL(SEG$(DATA$,POS(DATA$,"=",1)+1,10))
1230 IF IB(IC)=0 THEN 1170
1240 INPUT #1:LI(1,IC)
1250 GOTO 1170
1260 IF TOTO$<>"IEN" THEN 1310
1270 IE(IC)=VAL(SEG$(DATA$,POS(DATA$,"=",1)+1,10))
1280 IF IE(IC)=0 THEN 1170
1290 INPUT #1:LI(2,IC)
1300 GOTO 1170
1310 IF TOTO$<>"ECR" THEN 1350
1320 ECR=VAL(SEG$(DATA$,POS(DATA$,"=",1)+1,10))
1330 CALL LOAD(-24572,ECR)
```

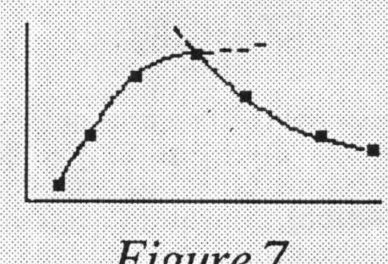


Figure 7

Pour éviter que ces discontinuités (les dérivées premières à droite et à gauche de chaque point de jonction peuvent très différentes) ne soient trop brutales, on peut décider de calculer le polynôme de degré 3 passant par quatre points voisins et *de ne garder cette équation que pour les deux points du milieu.* On décale alors d'un point et on renouvelle l'opération (voir figure 8).

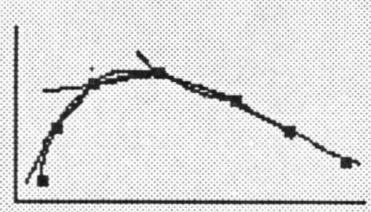


Figure 8

Pour un jeu de *m* points nous avons donc besoin de caluler non pas approximativement *m/3* équations mais *m-1*. En effet chaque intervalle de deux points se trouve décrit par une équation différente. Cette technique beaucoup plus longue en temps calcul que la précédente (la même équation pour les quatre points voisins) donne un aspect beaucoup plus homogène à la courbe. Le tracé ainsi calculé peut néanmoins toujours comporter des cassures.

Mathématiquement cela se traduit par le fait qu'aucune contrainte sur les dérivées premières à droite et à gauche pour chaque point n'a été formulée. Le calcul se fait simplement en calculant à chaque fois le polynôme d'ordre 3 passant par quatre points (système linéaire de 4 équations à 4 inconnues). Tous ces systèmes 4,4 sont indépendants et peuvent être résolus dans n'importe quel ordre.

Analysons maintenant la base d'une technique formulant certaines conditions sur les dérivées en chaque point.

Interpolation par 'splines' cubiques

Sans entrer dans le détail sur la théorie des 'splines', voyons quelle est le fil directeur de cette technique. Le qualificatif cubique indique ici que les polynômes d'interpolation seront de degré 3 (il existe des 'splines' de tout ordre).

L'idée est toujours de calculer un polynôme (de degré 3 donc) pour interpoler deux points mais non plus en le définissant comme la fonction passant par quatre points voisins (il faut 4 équations pour calculer les 4 paramètres du polynôme) mais comme le polynôme passant par deux points (2 équations) et dont les dérivées première et seconde à gauche du deuxième point sont égales aux dérivées première et seconde à droite du premier point du polynôme suivant (celui qui sera calculé sur l'intervalle suivant). Relisez bien cette phrase, elle est fondamentale pour bien comprendre la technique des splines. Ceci donne bien 2 nouvelles équations, le système est donc bien soluble.
On voit bien qu'il est impossible de calculer indépendamment chaque polynôme puisque chacun d'entre eux dépend de son suivant (et donc de son précédent). La résolution se fera donc globalement, sur l'ensemble de tous les points.

Géométriquement, cela se traduit par les propriétés suivantes :

- pas de cassures en un point (égalité des dérivées premières à droite et à gauche);
- pas d'inflexion sur un point (égalité des dérivées secondes à droite et à gauche).

Rappelons qu'un point d'inflexion correspond à un changement de signe du rayon de courbure de la fonction (voir figure 9).

```
1340 GOTO 1170
1350 IF TOTO$<>"NOU" T
     HEN 1510
1360 IF DP+(DC=0)THEN
     1380
1370 CP(IC)=CC(IC)
1380 IF (DP=0)+DC THEN
      1400
1390 CC(IC)=CP(IC)
1400 NP(IC)=P
1410 IC=IC+1
1420 P=0
1430 REM  valeurs par
     defaut
1440 CP(IC)=9
1450 CC(IC)=15
1460 IB(IC)=0
1470 IE(IC)=0
1480 DP=0
1490 DC=0
1500 GOTO 1170
1510 IF TOTO$<>"POI" T
     HEN 1550
1520 DP=1
1530 CP(IC)=VAL(SEG$(D
     ATA$,POS(DATA$,"=
     ",1)+1,10))
1540 GOTO 1170
1550 IF TOTO$<>"COU" T
     HEN 1590
1560 DC=1
1570 CC(IC)=VAL(SEG$(D
     ATA$,POS(DATA$,"=
     ",1)+1,10))
1580 GOTO 1170
1590 V=POS(DATA$," ",1
     )
1600 P=P+1
1610 A(P,IC)=VAL(SEG$(
     DATA$,1,V-1))
1620 B(P,IC)=VAL(SEG$(
     DATA$,V+1,10))
1630 U=A(P,IC)
1640 V=B(P,IC)
1650 IF P*IC-1 THEN 17
     00
1660 XMAX=U
1670 YMAX=V
1680 XMIN=U
```

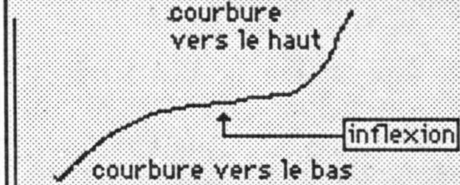


Figure 9

Les tracés ainsi obtenus, au prix de quelques efforts supplémentaires (entièrement assumés par notre TI), sont très réguliers et ne comportent aucune cassure. Pour des ensembles de points très "tourmentés", on obtient des tracés très "variants" mais toujours sans aucune discontinuité (voir figure 10).

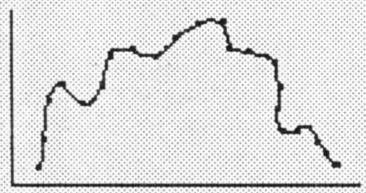


Figure 10

Pour vous en convaincre nous proposons ici un petit programme de démonstration permettant de visualiser un jeu de données, de faire une régression cubique par la méthode des moindres carrés et de faire un tracé au moyen de splines cubiques.

Utilisation du programme de démonstration

Ce programme permet donc de dessiner à l'écran soit une courbe moyenne (polynôme du troisième degré) ajustée à un jeu de données par la méthode des moindres carrés, soit une courbe exacte passant par tous les points et utilisant la technique des splines cubiques.

Le programme lit les données dans un fichier DIS/VAR 80 qui peut être créé par l'éditeur/assembleur ou par un programme basic. Le programme peut traiter simultanément plusieurs jeux de données.

Structure du fichier d'entrée

Le fichier contient les différents jeux de données à traiter et optionnellement divers mots-clés ayant chacun un rôle particulier. Chaque ligne de données doit comporter deux valeurs (xi et yi) séparées par un espace. Les mots-clés sont les suivants :

; signifie qu'il s'agit d'une ligne de commentaires qui doit donc être ignorée;

ECRAN= suivi d'un code (0 à 15) indique la couleur de l'écran.

Chaque jeu de données doit être séparé du suivant (si suivant il y a) par :

NOUVELLE COURBE

Enfin, chaque jeu de données peut optionnellemnt contenir :

POINTS= suivi du code (0 à 15) pour la couleur des points;

COURBE= suivi du code (0 à 15) pour la couleur du tracé.

Par défaut, le programme prend 1 comme couleur d'écran (noir), 9 pour celle des points (rouge) et 15 pour celle des courbes (blanc). Si vous définissez la couleur des points et pas celle de la courbe ou réciproquement, le programme prendra cette couleur pour les deux.

On peut ainsi superposer sur une même image graphique plusieurs courbes de couleurs différentes (avec les restrictions propres au mode bit-map du TI à savoir une seule couleur "on" et une seule couleur "off" pour chaque rangée de 8 points)

De plus, pour le tracé à l'aide de splines cubiques, le fichier peut contenir

IBEG= 0, 1 ou 2

et/ou

IEND= 0, 1 ou 2

qui déterminent les conditions aux limites du tracé. Si ces informations ne figurent pas dans le fichier, leurs valeurs par défaut sont 0.

```
1690 YMIN=V
1700 IF U>=XMIN THEN 1
     720
1710 XMIN=U
1720 IF U<=XMAX THEN 1
     740
1730 XMAX=U
1740 IF V>=YMIN THEN 1
     760
1750 YMIN=V
1760 IF V<=YMAX THEN 1
     780
1770 YMAX=V
1780 GOTO 1170
1790 NP(IC)=P
1800 IF DP+(DC=0)THEN
     1820
1810 CP(IC)=CC(IC)
1820 IF (DC=0)+DC THEN
      1840
1830 CC(IC)=CP(IC)
1840 CLOSE #1
1850 DEF FA(X)=245*(X-
     XMIN)/(XMAX-XMIN)
     +5
1860 DEF FO(X)=181-171
     *(X-YMAX)/(YMIN-Y
     MAX)
1870 FOR J=1 TO IC
1880 FOR I=1 TO NP(J)
1890 A(I,J)=FA(A(I,J))
1900 B(I,J)=FO(B(I,J))
1910 NEXT I
1920 NEXT J
1930 RETURN
1940 REM  afficher les
      points
1950 CALL CLEAR
1960 PRINT "Afficher l
     es points"
1970 FOR I=1 TO IC
1980 CALL LOAD(-24573,
     CP(I))
1990 FOR J=1 TO NP(I)
2000 CALL LOAD(-24576,
     A(J,I),B(J,I))
2010 CALL LINK("POINT"
     )
2020 NEXT J
2030 NEXT I
2040 GOSUB 730
2050 CALL LINK("IMAGE"
     )
2060 RETURN
2070 REM  Trace par mo
     indres carres
2080 CALL CLEAR
2090 PRINT "Trace par
     regression polyno
     -miale d'ordre 3"
     ::"Resolution et
```

IEND a le même rôle que IBEG. Le premier fixe la condition limite du dernier point, le second celle du premier point du jeu de données.

IBEG=0 (respectivement IEND=0) signifie que le polynôme d'interpolation pour les points 2 et 3 (respectivement *n-2* et *n-1*) doit également être utilisé pour le premier point (respectivement pour le dernier).

IBEG=1 (respectivement IEND=1) signifie que vous fournissez sur la ligne suivante du fichier la valeur de la dérivée *première* de la courbe au premier point (respectivement au dernier). L'enregistrement suivant du fichier doit donc contenir cette grandeur.

IBEG=2 (respectivement IEND=2) signifie que vous fournissez sur la ligne suivante du fichier la valeur de la dérivée *seconde* de la courbe au premier point (respectivement au dernier). L'enregistrement suivant du fichier doit donc contenir cette grandeur.

Le fichier de démonstration contient deux fois le même jeu de données avec des conditions aux limites différentes (cas des "splines"). Les "splines" ainsi obtenues diffèrent très sensiblement (voir figure 11) selon ces conditions. Cet exemple montre que la connaissance de ces valeurs peut influer fortement sur l'aspect de la courbe.

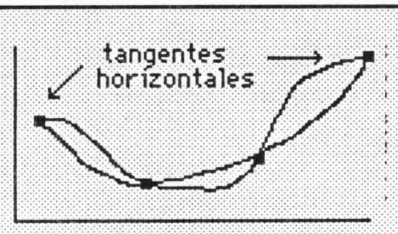


Figure 11

En l'absence de tout renseignement fiable sur l'allure de la courbe, il est préférable de prendre 0 pour IBEG et IEND, donc de ne rien mettre (valeurs par défaut).

Caractéristiques générales du programme

Le lancement du programme fait apparaitre un menu permettant :

1) de lire un jeu de données
Cette option vous demande le nom du fichier d'entrée. Vous pouvez simplement taper ENTER, le programme prendra alors le fichier **DSK1.DATA**. Si vous tapez juste un nom de fichier le programme rajoutera automatiquement "**DSK1.**". Vous avez de plus la possibilité de faire défiler à l'écran tous les fichiers de type DIS/VAR 80 du disque **i** (1,2 ou 3) en tapant CATi.

2) d'afficher les points
Le programme affiche alors simplement les différents jeux de données dans les couleurs choisies.

3) d'effectuer un tracé à base de splines cubiques
Le programme construit toutes les "splines" nécessaires à chaque tracé suivant les conditions limites spécifiées dans le fichier d'entrée, puis les affiche en même temps que les données du fichier.

4) d'effectuer un tracé par la méthode des moindres carrés
Une régression d'ordre 3 est effectuée sur chaque famille de points. Sont affichés sur l'écran tous les jeux de points et toutes les courbes calculées. On peut ainsi apprécier la finesse de l'interpolation.

5) d'effacer l'image graphique
L'image graphique n'est pas effacée à chaque nouveau tracé, vous permettant ainsi de superposer un tracé obtenu par les "splines" à un tracé de la même famille de points obtenu par la méthode des moindres

```
     trace en cours"
2100 FOR IS=1 TO IC
2110 FOR I=1 TO 4
2120 FOR J=1 TO 7
2130 C(I,J)=0
2140 NEXT J
2150 NEXT I
2160 N=NP(IS)
2170 CALL LOAD(-24573,
     CC(IS))
2180 FOR K=1 TO 6
2190 XR(K)=0
2200 FOR I=1 TO N
2210 XR(K)=XR(K)+A(I,I
     S)^K
2220 NEXT I
2230 NEXT K
2240 YC=0
2250 FOR I=1 TO N
2260 YC=YC+B(I,IS)
2270 NEXT I
2280 FOR K=1 TO 3
2290 FOR I=1 TO N
2300 C(K,7)=C(K,7)+B(I
     ,IS)*A(I,IS)^K
2310 NEXT I
2320 NEXT K
2330 FOR I=1 TO 4
2340 FOR J=1 TO 4
2350 IJ=I+J-2
2360 IF (I=1)*(J=1)THE
     N 2390
2370 C(I,J)=XR(IJ)
2380 GOTO 2400
2390 C(1,1)=N
2400 NEXT J
2410 NEXT I
2420 C(1,6)=YC
2430 FOR I=2 TO 4
2440 C(I,6)=C(I-1,7)
2450 NEXT I
2460 REM  calculs des
     coeffs
2470 FOR K=1 TO 3
2480 FOR I=K+1 TO 4
2490 C(I,6)=C(I,6)-C(I
     ,K)/C(K,K)*C(K,6)
2500 FOR J=K+1 TO 4
2510 C(I,J)=C(I,J)-C(I
     ,K)/C(K,K)*C(K,J)
2520 NEXT J
2530 NEXT I
2540 NEXT K
2550 C(4,5)=C(4,6)/C(4
     ,4)
2560 FOR I=3 TO 1 STEP
      -1
2570 S=0
2580 FOR K=I+1 TO 4
2590 S=S+C(I,K)*C(K,5)
2600 NEXT K
```

carrés. Pour effacer la page graphique avant un nouveau tracé, utilisez donc cette option.

Remarques :

Tous les jeux de données doivent être rentrés **abscisses croissantes.**

Une fois l'image graphique affichée, vous pouvez toujours utiliser les différents utilitaires décrits dans les numéros précédents à savoir :

CTRL-N pour une "hardcopy" normale;

CTRL-I pour une "hardcopy" en inversé (off ⇒ on et on ⇒ off);

CTRL-S pour sauvegarder l'image graphique sur disque;

CTRL-C pour charger une image graphique à partir d'un disque;

BACK pour revenir au Basic.

Ce programme ne décrit que le fonctionnement des "splines" et de la méthode des moindres carrés. Bien qu'il soit proche de saturer les 16 petits K-octets de notre TI (la dimension des tableaux permet de traiter jusqu'à 5 courbes, chacune définie par 15 points), vous pouvez toujours envisager de l'étendre en rajoutant par exemple :

- introduction de texte (routine "LETTRE") sur la page graphique;
- introduction d'axes ("AXES") et des principales valeurs
- etc…

Pour gagner un peu de mémoire, vous pouvez toujours :

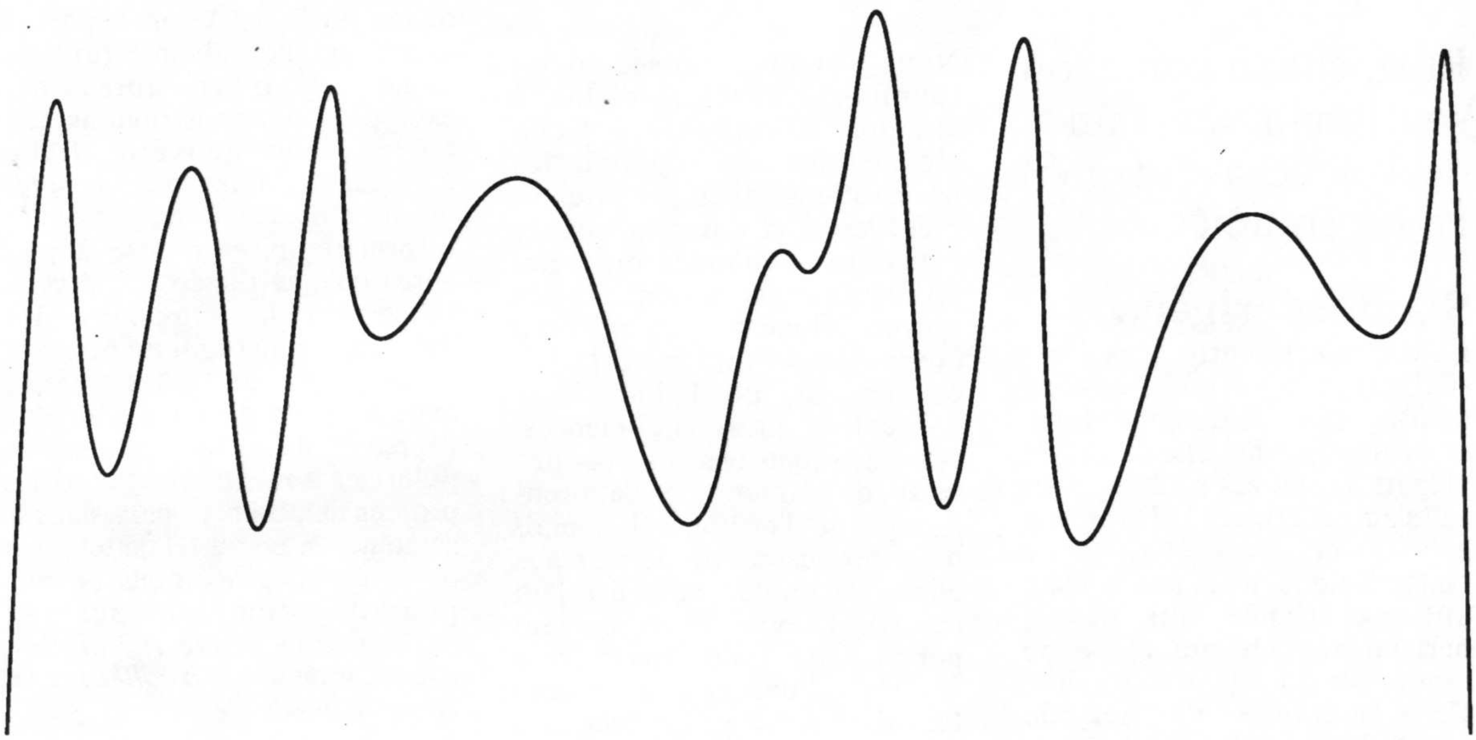
- supprimer les remarques;
- diminuer la taille des tableaux;
- faire CALL FILES(1) avant le chargement.

Bon Spleen ...

99

```
2610 C(I,5)=(C(I,6)-S)/C(I,I)
2620 NEXT I
2630 REM DEFINITION POLYNOME
2640 DEF R(X)=C(1,5)+X*(C(2,5)+X*(C(3,5)+X*C(4,5)))
2650 FOR I=A(1,IS)TO A(N,IS)
2660 CALL LOAD(-24568,I,R(I),I+1,R(I+1))
2670 CALL LINK("LIGNE")
2680 NEXT I
2690 NEXT IS
2700 GOSUB 1950
2710 RETURN
2720 REM  Trace par splines cubiques
2730 CALL CLEAR
2740 PRINT "Trace par Splines cubiques"::"Resolution et trace en cours"
2750 REM   Resolution
2760 FOR IS=1 TO IC
2770 N=NP(IS)
2780 CALL LOAD(-24573,CC(IS))
2790 REM  remplissage matrice
2800 FOR I=1 TO N
2810 C(1,I)=B(I,IS)
2820 NEXT I
2830 REM   conditions limites
2840 IBEG=IB(IS)
2850 IEND=IE(IS)
2860 C(2,1)=LI(1,IS)
2870 C(2,N)=LI(2,IS)
2880 REM  resolution
2890 L=N-1
2900 FOR M=2 TO N
2910 C(3,M)=A(M,IS)-A(M-1,IS)
2920 IF C(3,M)THEN 2940
2930 C(3,M)=.001
2940 C(4,M)=(C(1,M)-C(1,M-1))/C(3,M)
2950 NEXT M
2960 ON IBEG+1 GOTO 2970,3060,3090
2970 IF N>2 THEN 3020
2980 C(4,1)=1
2990 C(3,1)=1
3000 C(2,1)=2*C(4,2)
3010 GOTO 3330
3020 C(4,1)=C(3,3)
3030 C(3,1)=C(3,2)+C(3,3)
3040 C(2,1)=((C(3,2)+2*C(3,1))*C(4,2)*C(3,3)+C(3,2)^2*C(4,3))/C(3,1)
3050 GOTO 3130
3060 C(4,1)=1
3070 C(3,1)=0
3080 GOTO 3120
3090 C(4,1)=2
3100 C(3,1)=1
3110 C(2,1)=3*C(4,2)-C(3,2)/2*C(2,1)
3120 IF N=2 THEN 3330
3130 FOR M=2 TO L
3140 G=-C(3,M+1)/C(4,M-1)
3150 C(2,M)=G*C(2,M-1)+3*(C(3,M)*C(4,M+1)+C(3,M+1)*C(4,M))
3160 C(4,M)=G*C(3,M-1)+2*(C(3,M)+C(3,M+1))
3170 NEXT M
3180 REM
3190 ON IBEG+1 GOTO 3200,3400,3290
3200 IF (N=3)*(IBEG=0) THEN 3260
3210 G=C(3,N-1)+C(3,N)
3220 C(2,N)=((C(3,N)+2*G)*C(4,N)*C(3,N-1)+C(3,N)^2*(C(1,N-1)-C(1,N-2))/C(3,N-1))/G
3230 G=-G/C(4,N-1)
3240 C(4,N)=C(3,N-1)
3250 GOTO 3380
3260 C(2,N)=2*C(4,N)
3270 C(4,N)=1
3280 GOTO 3370
3290 C(2,N)=3*C(4,N)+C(3,N)/2*C(2,N)
3300 C(4,N)=2
3310 GOTO 3370
3320 REM
3330 ON IEND+1 GOTO 3340,3400,3290
3340 IF IBEG>0 THEN 3260
3350 C(2,N)=C(4,N)
3360 GOTO 3400
3370 G=-1/C(4,N-1)
3380 C(4,N)=G*C(3,N-1)+C(4,N)
3390 C(2,N)=(G*C(2,N-1)+C(2,N))/C(4,N)
3400 FOR J=L TO 1 STEP -1
```

```
3410 C(2,J)=(C(2,J)-C(
     3,J)*C(2,J+1))/C(
     4,J)
3420 NEXT J
3430 FOR I=2 TO N
3440 DTAU=C(3,I)
3450 DF1=(C(1,I)-C(1,I
     -1))/DTAU
3460 DF3=C(2,I-1)+C(2,
     I)-2*DF1
3470 C(3,I-1)=2*(DF1-C
     (2,I-1)-DF3)/DTAU
3480 C(4,I-1)=(DF3/DTA
     U)*(6/DTAU)
3490 NEXT I
3500 REM  fin de resol
     ution
3510 FOR I=1 TO N-1
3520 REM  construction
      polynome pour ch
     aque intervalle
3530 DEF PN(X)=C(1,I)+
     (X-A(I,IS))*(C(2,
     I)+(X-A(I,IS))*(C
     (3,I)+(X-A(I,IS))
     *C(4,I)/3)/2)
3540 FOR J=A(I,IS)+1 T
     O A(I+1,IS)
3550 CALL LOAD(-24568,
     J-1,PN(J-1),J,PN(
     J))
3560 CALL LINK("LIGNE"
     )
3570 NEXT J
3580 NEXT I
3590 NEXT IS
3600 GOSUB 1970
3610 RETURN
3620 REM  Sortie du pr
     ogramme
3630 CALL CLEAR
3640 END
```



```
;FICHIER DE DEMONSTRATION
;
;le meme jeu de donnees est utilise
;avec des conditions limites differentes
;
;Ecran noir par defaut
;courbe blanche par defaut
;avec points rouges
;
;pas de valeur pour IBEG et IEND
;donc 0 pour les deux
;
10 10
20 5
30 5
40 10
50 25
NOUVELLE COURBE
;meme jeu de donnees
COURBE=11
;courbe jaune
POINTS=9
;maintien de la couleur rouge
;    pour les points
IBEG=1
0
;derivee premiere nulle c-a-d
;tangente horizontale pour le
;premier point
IEND=1
0
;idem pour le dernier point
10 10
20 5
30 5
40 10
50 25
```

Expérience pédagogique : orthographe & informatique

Philippe Misandeau

Et si, sur sa copie, on soulignait les fautes de l'élève avant qu'il ne les commette…

S'autocorriger…

L'idée n'est certe pas très réalisable… Nous savons pourtant que, si, de notre doigt, nous soulignons une faute, la plupart des élèves sont capables se s'autocorriger ! Dans cet esprit, et avec l'aide de l'informatique, nous avons tenté une expérience qui devait permettre à chaque élève de minimiser sa disorthographie. Cette méthode n'a rien de miraculeux, elle veut juste permettre à l'élève de focaliser son attention sur les points précis qui sont la source de ses mauvaises notes en orthographe et cela, bien sûr, avant notre propre correction.

Niveau et conditions

Cette expérience a été menée en classe de 5ème et 4ème, sur 5 dictées empruntées au manuel scolaire en cours dans l'établissement, ou dans diverses "méthodes d'orthographe", toutes plus prometteuses" les unes que les autres.

Des fiches d'observations

Nous avons pensé qu'en fournissant avant la dictée, à chaque élève, une fiche signalétique qui reportait les observations faites sur sa copie précédente, et qui résumait ses mauvaises habitudes orthographiques, nous avions là un moyen efficace de le prévenir contre ses fautes et, par conséquent, de l'aider à se concentrer sur ses déficiences. Nous ne proposons donc pas une méthode directe de l'apprentissage de l'orthographe, mais nous proposons de donner aux élèves l'habitude de relire leur travail, l'envie d'exercer leur perspicacité à découvrir leurs "fautes". Pour cela nous leur fournissons un guide adapté à leur profil, afin qu'il puissent facilement prendre leur orthographe en charge, et que par réflexe, ils sachent où porter leur regard lors de la relecture de leur copie.

En tout premier lieu, il fallait établir une échelle commune, représentative des erreurs les plus fréquemment rencontrées. Nous avons retenu onze rubriques pour réaliser ces fiches-guide individuelles. *Voir en encadré.*

Sur cette fiche, nous rédigeons, à la main, nos observations relatives à la correction de la dictée du jour. Avant l'épreuve d'orthographe suivante (une ou deux semainbes après par exemple), nous distribuons aux élèves leur nouvelle fiche personnelle, imprimée, proprement "sortie" du fichier informatique, et remise à jour après chaque dictée. C'est cette nouvelle fiche augmentée des derniers renseignements sur l'orthographe de l'enfant, de ses éventuels progrès qui, lors de l'épreuve, devra lui permettre de polariser son attention sur ses propres défaillances puis, dans la semaine, de revoir tel ou tel point de sa grammaire : nous devons pouvoir compter sur sa participation active… *Voir les exemples de fiches à coller sur la copie, avant dictée.*

De l'avantage de l'ordinateur

L'informatique n'avait rien ici d'indispensable : nous pouvions

1	**Nom prénom**	de l'élève
2	**Accord**	sujet-verbe, participes…
3	**Accord**	nom-adjectif…
4	**Conjugaison**	pers, tps, modes à revoir…
5	**Confusion de sonorités**	ai-er se-ze ou-au…
6	**Vocabulaire**	nombre de fautes
7	**Ponctuation, accentuation**	
8	**Remarques diverses**	lisibilité…
9	**Meilleure note obtenue**	
10	**Dernière note obtenue**	
11	**Progrès**	o/n depuis la dernière dictée

Rubriques retenues

FICHE 13	
Nom	*Jackson Jean*
Accord	*nom/adjectif*
Accord	*verbe/sujet*
Conjug.	*passé simple*
Sonorité	
Vocabul.	
Ponctuat.	
Divers	*bien relire*
Note Max.	*10,00*
Dern. note	*10,00*
Progrès	*beaucoup*

FICHE 14	
Nom	*Janvier Noël*
Accord	*nom/adjectif*
Accord	*verbe/sujet*
Conjug.	
Sonorité	
Vocabul.	
Ponctuat.	
Divers	*écriture*
Note Max.	*8,00*
Dern. note	*2,00*
Progrès	*en baisse*

Exemples de fiches à coller

très bien tenir à jour, et à la main, les fiches des élèves. Mais, le système informatique représente quelques avantages certains. Cela demande bien sur une mise en œuvre assez complexe : un ordinateur avec lecteur de disquettes, une imprimante, un logiciel qui permet la mémorisation rapide des informations spécifiques aux rubriques concernées. Il faut ajouter au temps de la correction traditionnelle le temps de saisie des données ! Mais, si le programme est bien pensé*, cela est encore assez rapide, de toutes les façons, plus rapide que le même travail effectué "à la main". En bénéfice, il faut retenir qu'à l'objet informatique qui n'a pas manqué d'intéresser les élèves, donc de les motiver dans leurs efforts (nous y reviendrons), s'est ajouté la grande lisibilité des fiches distribuées, plus efficaces sous cette forme que nos appréciations trop souvent écrites à la hâte (et si vite oubliées entre deux épreuves !). En cas de disorthographie profonde, au lieu de surcharger la fiche de l'élève, nous avons préféré l'aider en le guidant sur plusieurs séances avec la même rubrique (accord verbe sujet par exemple), afin de ne pas disperser son attention sur tous les écueils qu'il lui faut vaincre.

Évaluation

Tout d'abord soulignons l'intérêt porté à la méthode par l'ensemble de la classe. Tous se sont volontiers pliés à la distribution des fiches avant la dictée, puis au collage de cette fiche sur leur copie (il n'y a eu aucun manquement sur toute la durée de l'expérience). Cependant, il n'a pas été très facile dévaluer en toute objectivité l'ensemble des résultat. Trois paramètres peuvent radicalement modifier notre évaluation. Cette approche a manifestement intéressé les élèves parce qu'il y avait de l'ordinateur "dans l'air" (phénomène de mode non négligeable). Tous savaient encore que cela déboucherait sur un compte-rendu d'expérience : pour quelle part ces deux points entrèrent dans léur désir de mieux faire ? Cela était difficle à comptabiliser ! Enfin, le troisième paramètre est lié à la durée de l'expérience : celle-ci ne s'est déroulée que sur cinq dictées, période de plus mise à profit pour remodeler tel ou tel point de l'expérimentation en cours. Mais, notons aussi que des élèves ont regretté l'absence de ces fiches lors de la toute dernière épreuve, verbalisant que ce guide leur manquait, et qu'ils désiraient vraiment voir se prolonger la méthode pour l'an prochain. *Voir les quelques chiffres évocateurs en encadré.*

* Pour cette expérience, un module "Gestion de Fichiers",

Dictée n°	*Elèves en progrès* O	N	*Erreurs sur accords du part. passé*	*Moyenne de la classe*	*Note maxi*	*Nombre de 0*
4ème						
1			22/29	1,2/20	12/20	12
2	17	12	20/29	4,9/20	19/20	10
3	10	19	17/29	4,8/20	16/20	10
4	19	10	10/29	6,8/20	17/20	07
5ème						
1			23/30	5,1/20	17,5/20	12
2	18	12	23/30	6,2/20	15,5/20	10
3	17	13	10/30	5,3/20	18,5/20	13
4	28	02	11/30	7,8/20	20/20	06
5	21	09	00/30	8,1/20	20/20	06

Quelques chiffres...

adapté au TI-99/4A qui était à notre disposition, nous a évité le temps consacré à la conception du programme.

Exemple de programme Pédagogique

Thème

Révision ou acquisition globale de la nature des mots qui composent la phrase, entrainement à la syntaxe sur les bases d'un jeu grammatical.

Niveau

Pour les élèves en C.M. et, en classe de soutien 6 ème et 5 ème. Ce travail nécessite, au préalable, une connaissance au moins approximative des divers constituants de la phrase : noms, adjectifs, adverbes, verbes, pronoms, articles, déterminant... Cependant, les enseignants peuvent modifier la difficulté de l'exercice, en introduisant, dans le programme et en DATA, des phrases plus ou moins complexes, voire mieux adaptées au niveau de leur classe (*cf* Modifications).

Objectifs pédagogiques

A la suite de l'utilisation de ce logiciel, l'enfant devra pouvoir reconnaitre la nature de chacun des mots d'une proposition, et savoir reconstruire une phrase à partir de ses éléments constitutifs, proposés en ordre logique ou illogique à sa sagacité. Pour maintenir son attention et sa réflexion en éveil, nous lui proposons un match où gagnera celui qui possède au mieux sa grammaire (*cf* évaluation des points).

Fonctionnement et commentaires

L'adulte a une part active dans le fonctionnement de ce programme qui se veut bref et facile à "entrer" dans la machine : il devra parfois expliquer une règle de grammaire, approfondir tel ou tel aspect de la syntaxe française, soutenir les élèves dans leur effort de réflexion... et faire "tourner" le programme :

A] L'ordinateur demande tout d'abord combien de groupes ou d'élèves vont "jouer", puis le nom des participants. (Expérience faite, il est souhaitable de n'avoir à contrôler que trois ou quatre productions, afin de ne pas démobiliser l'attention des enfants).

B] Sur l'écran (vide de toute autre information), s'affichent une à une, pendant un temps modulable (boucle K en ligne 75 et 80), les natures des différents paradigmes qui devront constituer la phrase à créer.

C] L'enfant doit noter ces renseignements puis, il cherchera à réaliser sur sa feuille une phrase à l'aide de toutes les natures de mots que l'ordinateur lui impose.

D] Il peut, en cas de besoin, venir consulter des fichiers afin d'obtenir des renseignements complémentaires à sa reflexion. Voici le menu affiché :

1 Classe des adjectifs (qualificatifs ou non qualificatifs)...
2 Ensemble des déterminants (articles ou adjectifs)
3 Pronoms
4 Adverbes
5 Conjonctions et prépositions
6 Solution du problème et entrée des points (fin du temps de réflexion)
7 Fin de travail et affichage du score final.

E] Au bout du temps de réflexion déterminé (en fonction de la difficulté ou par exemple, après que trois élèves aient trouvé une solution), l'adulte commente et évalue le résultat écrit par chacun puis, en appuyant sur la touche 6, il affiche à l'écran la phrase type, solution au problème posé : un dernier commentaire grammatical qui justifie rapidement la solution proposée sera le bienvenu.

F] La machine est maintenant prête pour une nouvelle expérience (12 ici).

Guide pour une évaluation

(A modifier en fonction du projet pédagogique et de la classe concernée). Dans cet exemple la consultation du fichier grammatical est intentionnellement freinée afin d'éviter la ruée vers l'unique machine... Le barème qui attribuera ses points à l'élève est à envisager de la façon suivante (à titre indicatif) :

- Phrase sensée (+/-) 2 points,
- Phrase grammaticale (+/-) 3 points,
- Ajout d'un mot non prévu - 2 points par mot
- Oubli d'une nature prévue - 5 points
- Consultation du fichier - 4 points.

Modifications possibles

Pour rendre la leçon plus performante et plus efficace à long terme, il est souhaitable de modifier quelque peu le programme. Il est possible par exemple :

- de varier la difficulté des phrases afin de mieux adapter celles-ci au niveau des élèves (*cf* DATA en ligne 400), ou d'introduire un choix aléatoire de ces phrases, préalablement classées en DATA selon leur niveau de difficulté.
- On peut encore imposer à la phrase finale, l'ordre d'apparition sur l'écran de ses différents constituants : respect d'un schéma syntaxique donné...
- Afficher la naturer des mots, accompagnée d'une musique propre, ou mieux, faire annoncer par la machine les différentes natures qu'elle propose...
- Si la classe possède autant d'ordinateurs que d'élèves, le calcul des points peut être automatisé, chacun entrant sur sa machine la phrase qu'il propose : ceci complique bien le programme, mais le transforme en véritable didacticiel. Ce n'était bien sur pas notre projet ici.

Programme "Grammaire"
Basic étendu

```
1 !********** PROGRAMME POUR TI 99
  /4A + BASIC ETENDU ****PH.M.*
10 !**** "   SOUTIEN DE GRAMMAIRE"
   ****
20 REM ***************************
   *****
25 CALL CLEAR :: CALL SCREEN(11)::
   DIM PH$(15)
30 RESTORE 490 :: FOR J=1 TO 7 ::
   READ L(J),C(J),M$(J):: NEXT J
40 INPUT "NBRE D'ELEVES OU DE GRP
   ":NG
50 FOR I=1 TO NG :: PRINT :"NOM DU
   GROUPE";I;"? ";:: INPUT "":NG$
   (I):: NEXT I
60 CALL CLEAR :: RESTORE 400
70 FOR I=1 TO 12 :: READ PH$(I),A
   :: CALL SOUND(200,196,0):: CALL
   SOUND(100,330,0):: CALL SOUND(
   200,262,0)
75 DISPLAY AT(10,13)ERASE ALL SIZE
   (14):"proposition";I :: DISPLAY
   AT(17,1):"faire une phrase ave
   c les": :"    elements proposes
   ..." :: FOR K=1 TO 1000 :: NEXT
   K
80 FOR J=1 TO A :: READ NAT$ :: DI
   SPLAY AT(10,6)BEEP ERASE ALL:TA
   B(12);J: :TAB(5);NAT$ :: FOR K=
   1 TO 1000 :: NEXT K :: NEXT J
100 CALL CLEAR :: FOR J=1 TO 7 ::
    DISPLAY AT(L(J),C(J)):M$(J)::
    NEXT J
110 DISPLAY AT(24,1):"VOTRE CHOIX?
      1-2-..-7"
120 CALL KEY(5,S,T):: IF S<49 OR S
    >55 OR T=0 THEN 120 ELSE S=S-4
    8
130 ON S GOTO 500,600,700,800,900,
    1000,1030
400 DATA JE TE SALUE BIEN,4,VERBE,
    ADVERBE,PRONOM PERS.,PRONOM PE
    RS.,QUAND PARS-TU?,3,VERBE,CON
    JONCTION,PRONOM PERS
405 DATA L'HIPPOPOTAME SE PROMENAI
    T  SUR UN FIL,6,VERBE,DETERMIN
    ANT,ARTICLE,NOM,NOM,PREPOSITIO
    N
410 DATA LE GENERAL A PRONONCE UN
       DISCOURS MEMORABLE,6,VERBE,
    ARTICLE,ARTICLE,NOM,NOM,ADJECT
    IF QUALIFICATIF
415 DATA PIERRE ET LA PETITE MARIE
     SONT DEUX TRES BEAUX ENFANT
    S,10,VERBE,ARTICLE,NOM,NOM,NOM
    ,ADJECTIF,ADJECTIF,ADJECTIF NO
    N-QUALI.,ADVERBE
420 DATA CONJONCTION,LUCKY LUKE PO
    URSUIVAIT LES 4 DALTON DEPUIS
    LONGTEMPS,7,VERBE,ARTICLE,NOM,
    NOM,ADJ. NON-QUALIFICATIF,ADVE
    RBE,PREPOSITION
425 DATA IL LE CONDUIT AU PENITENC
    IER,5,VERBE,DETERMINANT,NOM,PR
    ONOM PERSONNEL,PRONOM,REGARDEZ
     OU VOUS METTEZ LES PIEDS,6,VE
    RBE,VERBE
430 DATA ARTICLE DEFINI,NOM,PRONOM
     PERSONNEL,PRONOM,IL TROUVERA
    PUISQU'IL CHER- CHE BEAUCOUP,6
    ,VERBE,VERBE,PRONOM,PRONOM,ADV
    ERBE,CONJONCTION
435 DATA IL FAUT QUE VOUS ME COMPR
    E- NIEZ BIEN,7,VERBE,VERBE,PRO
    NOM PERSONNEL,PRONOM,PRONOM PE
    RSONNEL,ADVERBE,CONJONCTION DE
     SUBORDINATION
440 DATA "UN BRUIT EFFROYABLE, ASS
    OUR-DISSANT: TROIS CENTS TONNE
    S QUI RETOMBENT BRUTALEMENT",1
    0,VERBE,ARTICLE,DETERMINANT,AD
    J. NON-QUALIFICATIF
445 DATA NOM,NOM,ADJECTIF QUALIFIC
    ATIF,ADECTIF,ADVERBE,PRONOM RE
    LATIF,CES   ELEVES   PATIENTS
           ATTENDENT  LA  SORTIE,
    6,VERBE,ARTICLE DEFINI
450 DATA NOM,NOM,ADJECTIF DEMONSTR
    ATIF,ADJECTIF QUALIFICATIF
490 DATA 5,2,1- ADJECTIFS,7,2,2- D
    ETERMINANTS,9,2,3- PRONOMS,11,
    2,4- ADVERBES,13,2,5-CONJONCTI
    ONS/PREPOSITIONS
495 DATA 15,2,6- SOLUTION ET SCORE
    ,17,2,7- FIN D'UTILISATION
500 DISPLAY AT(1,4)ERASE ALL:"** L
    ES ADJECTIFS **"
505 DISPLAY AT(3,1):"** QUALIFICAT
    IFS:                ...BLEU, B
    LANC,ROUGE..."
506 DISPLAY AT(6,1):"** DETERMINAT
    IFS (6 ESPECES)": :"1-NUMERAUX
    : ...UN, DEUX,          DOUBLE,
    CINQUIEME..."
510 DISPLAY AT(11,1):"2-POSSESSIFS
     : ...MON, NOTRE,    SES, LEUR,
     VOS, TON..."
515 DISPLAY AT(14,1):"3-DEMONSTRAT
    IFS:   CET, CES
       CE, CETTE"
520 DISPLAY AT(17,1):"4-RELATIFS:
```

```
(RARES)              .. LEQUEL m
onument est .."
525 DISPLAY AT(20,1):"5-INTERROGAT
IFS, EXCLAMATIFS      ...QUELLE.
..?, QUELS...!"
530 DISPLAY AT(23,1):"6-INDEFINIS:
...CERTAIN, NUL     PLUSIEURS,
AUTRE...";
535 CALL KEY(5,S,T):: IF T=0 THEN
535 ELSE 100
600 DISPLAY AT(1,4)ERASE ALL:"***
DETERMINANTS ***"
605 DISPLAY AT(4,1):"1-ADJ. NON-QU
ALIFICATIFS:          NUMERAUX, P
OSSESSIFS..."
610 DISPLAY AT(11,1):"2-ARTICLES D
EFINIS:                LE, LA, LE
S, AU, DU"
615 DISPLAY AT(14,1):"3-ARTICLES I
NDEFINIS:                   UN,
UNE,     DES "
620 DISPLAY AT(17,1):"4-ARTICLES P
ARTITIFS:                DU,     DE
LA,      DES "
625 DISPLAY AT(22,1):"---APPUYER S
UR 1 TOUCHE-----";
640 CALL KEY(5,S,T):: IF T=0 THEN
640 ELSE 100
700 DISPLAY AT(1,4)ERASE ALL:"***
LES PRONOMS   ***"
705 DISPLAY AT(3,1):"1-PRONOMS PER
SONNELS: ...JE,     ME, MOI,NOU
S, VOUS, LE.."
710 DISPLAY AT(6,1):"2-PR. POSSESS
IFS                    ...LE MIEN
, LES LEUR..."
715 DISPLAY AT(9,1):"3-PR. DEMONST
RATIFS:               ...CELUI, C
ELUI-CI, ..."
720 DISPLAY AT(12,1):"4-PR. RELATI
FS: LEQUEL, QUI,    QUE, QUOI,
DONT, OU..."
725 DISPLAY AT(15,1):"5-INTERROGAT
IFS:                     ..., QUI..
.?, QUE...?,.."
730 DISPLAY AT(18,1):"6-PR. INDEFI
NIS: .AUCUN, NUL     AUTRUI, PE
RSONNE, L'UN.."
735 DISPLAY AT(21,1):"7-PR. ADVERB
IAUX:                  EN, Y (mis
pour animal.)";
740 DISPLAY AT(24,1):"---APPUYER S
UR 1 TOUCHE----";
745 CALL KEY(5,S,T):: IF T=0 THEN
745 ELSE 100
800 DISPLAY AT(1,4)ERASE ALL:"***
LES ADVERBES ***"
805 DISPLAY AT(4,1):"1-ADV. DE MAN
IERE: ...,-MENT     PLUS,EXPRES
,MAL,VITE,..."
810 DISPLAY AT(7,1):"2-ADV. D'INTE
NSITE,QUANTITE:       ..ASSEZ, AU
TANT, MOINS,       PEU, TRES, CO
MBIEN, ..."
815 DISPLAY AT(11,1):"3-ADV. DE TE
MPS:  ...ALORS,       BIENTOT,AU
JOURD'HUI, TOT    AVANT, MAINT
ENANT,..."
820 DISPLAY AT(15,1):"4-ADV. LIEU:
...AILLEURS,          DEDANS, ICI
, LA, LOIN,..."
825 DISPLAY AT(18,1):"5-ADV. AFFIR
MATION:  ...OUI,      CERTES, SI
, BIEN, SOIT.."
830 DISPLAY AT(21,1):"6-ADV. DE NE
GATION:  NON, NE"
835 DISPLAY AT(23,1):"7-ADV. DE DO
UTE:    ...PROBA-   BLEMENT, SA
NS DOUTE, ...";
845 CALL KEY(5,S,T):: IF T=0 THEN
845 ELSE 100
900 DISPLAY AT(1,4)ERASE ALL:"***
CONJONCTIONS ***"
910 DISPLAY AT(6,1):"1-CONJ. DE CO
ORDINATION:          ET,OU,NI,MAI
S,CAR,OR,DONC    ENSUITE, PUIS,
AINSI..."
920 DISPLAY AT(11,1):"2-CONJ. DE S
UBORDINATION:        SI, SINON,
COMME, QUAND,      QUE, SI BIEN
QUE, ..."
925 DISPLAY AT(18,4):"*** PREPOSIT
IONS ****"
930 DISPLAY AT(21,1):"A, APRES, AV
EC, CHEZ, CONTRE DANS, DE, DEP
UIS, DURANT,    EN, ENTRE, SUR,
SANS, POUR";
945 CALL KEY(5,S,T):: IF T=0 THEN
945 ELSE 100
1000 CALL CLEAR :: PRINT "AFFICHAG
E D'UNE SOLUTION:": : : :PH$(
I): : : : : :"CALCUL DES POIN
TS:": :
1010 FOR J=1 TO NG :: PRINT NG$(J)
;PT(J);"+";
1011 INPUT PT1
1015 PT(J)=PT(J)+PT1 :: NEXT J ::
FOR K=1 TO 100 :: NEXT K
1020 NEXT I
1030 CALL CLEAR :: FOR I=1 TO NG :
: PRINT NG$(I),PT(I): : :: N
EXT I
1100 END
```

Georges Goument

Barmélo(s)

En voir de toutes les couleurs : tel pourrait bien être le but des deux jeux proposés ci-dessous. Avant de devenir des passe-temps, ils seront probablement des casse-têtes…

Barmélo 1

Il s'agit d'un jeu de réflexion basé sur un système de permutation de 5 couleurs sur un plateau de 25 cubes.

Le principe

Votre TI choisit une couleur pour un cube qu'il va disposer 25 fois sur l'écran. Il permuttera les couleurs entre 10 et 20 fois.

Vous vous retrouvez avec des cubes multicolores. Le but est bien sûr de redonner à chacun sa couleur d'origine.

Utilisation du clavier

E, X, S, D
Déplacement d'un cube à l'autre de la petite croix clignotante ; cette croix détermine le cube qui sert de base à la permutation.

ENTER
Effectue la permutation de couleur selon le principe ci-dessous :
Le cube joué passe à la couleur de valeur inférieure, les 2, 3 ou 4 contigus passent à la couleur de valeur supérieure.

V
Compare le cube que vous proposez avec le cube d'origine, impossible de tricher…

I
Pour interrompre la partie.

Vous n'aurez aucune difficulté pour suivre l'évolution du jeu, de nombreux commentaires étant insérés dans le programme.

Théoriquement, le nombre maximum de coups à jouer pour remettre la bonne couleur en place est de 100. Votre TI, très compréhensif vous en accordera 150, sachez en profiter…

Barmélo 2

Même principe, mais ce jeu introduit la troisième dimension pour garantir la migraine. Nous trouvons ici 48 cubes en 6 couleurs disposés sur 3 plateaux.

Le but

Redonner aux 48 cubes du jeu une couleur identique à celle choisie par l'ordinateur.

Le clavier

1, 2, 3
Choix du plateau (un cube clignote)

E, X, S, D
Sélectionne l'un ou l'autre des cubes : ils clignotent à tour de rôle.

ENTER
Réalise la permutation (voir Barmélo 1)

V, I
Mêmes fonctions que pour Barmélo 1

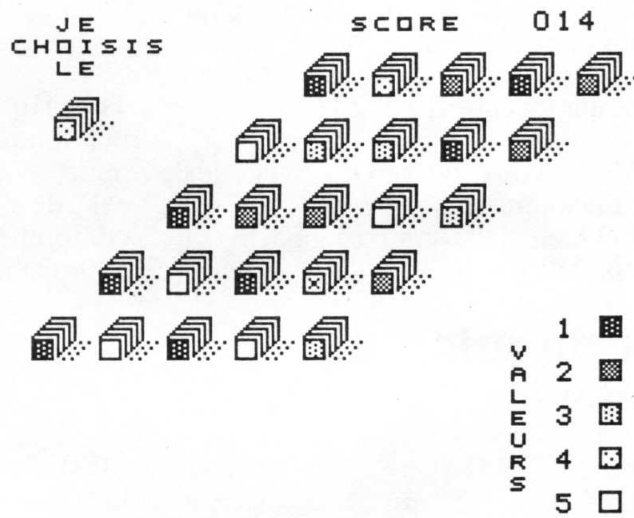
Attention, ce jeu vous réserve une petite complication supplémentaire…

99

Barmélo 1

```
10 REM *************
11 REM *           *
12 REM * BARMELO 1 *
13 REM *           *
14 REM * basic  TI *
15 REM *           *
16 REM * copyright *
17 REM *           *
18 REM *99 Magazine*
19 REM *           *
20 REM *G.  GOUMENT*
21 REM *           *
22 REM *************
23 CALL CLEAR
24 IN=0
25 PG=0
26 NJ=0
27 FOR I=1 TO 12
28 CALL COLOR(I,1,1)
29 NEXT I
30 CALL CHAR(64,"0")
31 CALL SCREEN(7)
32 PRINT " georges"::"
       goument":::TAB(
   9);"vous propose"::
   :::TAB(9);"@@@@@@@@
   @@@@":TAB(9);"@BARM
   ELO@@1@"
33 PRINT TAB(9);"@@@@@
   @@@@@@@"::::TAB(7)
   ;"jeu de reflexion"
   :::TAB(7);"niveau
   debutant"
34 FOR I=1 TO 12
35 CALL COLOR(I,15,1)
36 NEXT I
37 CALL COLOR(3,12,13)
38 FOR I=5 TO 8
39 CALL COLOR(I,12,13)
40 NEXT I
41 GOSUB 523
42 CALL CLEAR
43 PRINT "patientez qu
   elques secondes"
44 FOR I=1 TO 200
45 NEXT I
46 CALL HCHAR(23,3,32,
   28)
47 FOR I=1 TO 12
48 CALL COLOR(I,1,1)
49 NEXT I
```

```
50 RANDOMIZE
51 CALL SCREEN(2)
52 CU=48
53 CD=48
54 CE=48
55 VE=0
56 RESTORE 61
57 FOR I=1 TO 5
58 READ A
59 CALL CHAR(A,"FFFFFF
   FFFFFFFFFF")
60 NEXT I
61 DATA 40,33,64,72,80
62 RESTORE 67
63 FOR I=1 TO 5
64 READ A
65 CALL CHAR(A,"010207
   081F207F8")
66 NEXT I
67 DATA 41,34,65,73,81
68 RESTORE 73
69 FOR I=1 TO 5
70 READ A
71 CALL CHAR(A,"FF01FD
   05F515D555")
72 NEXT I
73 DATA 42,35,66,74,82
74 RESTORE 79
75 FOR I=1 TO 5
76 READ A
77 CALL CHAR(A,"555654
   5950644092")
78 NEXT I
79 DATA 43,36,67,75,83
80 RESTORE 85
81 FOR I=1 TO 5
82 READ A
83 CALL CHAR(A,"004800
   20008")
84 NEXT I
85 DATA 44,37,68,76,84
86 CALL CHAR(88,"00002
   4181824")
87 PRINT TAB(25);"1 !"
   :TAB(23);"v":TAB(23
   );"a 2 (":TAB(23);"
   l":TAB(23);"e 3 @":
   TAB(23);"u":TAB(23)
   ;"r 4 H"
88 PRINT TAB(23);"s":T
   AB(25);"5 P"
89 M$="score    000"
90 U=1
91 V=17
92 GOSUB 470
93 CALL COLOR(1,12,1)
94 CALL COLOR(2,6,1)
95 CALL COLOR(3,8,1)
96 CALL COLOR(4,8,1)
97 CALL COLOR(5,9,1)
98 CALL COLOR(6,13,1)
99 CALL COLOR(7,16,1)
100 FOR I=9 TO 12
101 CALL COLOR(I,8,1)
102 NEXT I
103 GOSUB 389
104 CC=INT((80-40+1)*R
    ND)+40
105 CALL SOUND(11,4600
    ,10)
106 IF CC=40 THEN 112
107 IF CC=33 THEN 112
108 IF CC=64 THEN 112
109 IF CC=72 THEN 112
110 IF CC=80 THEN 112
111 GOTO 104
112 CALL HCHAR(6,5,CC)
113 CALL HCHAR(6,6,CC+
    3)
114 CALL HCHAR(6,7,CC+
    4)
115 CALL HCHAR(5,5,CC+
    1)
116 CALL HCHAR(5,6,CC+
    2)
117 X1=16
118 X2=28
119 Y=4
120 GOSUB 140
121 X1=13
122 X2=25
123 Y=7
124 GOSUB 140
125 X1=10
126 X2=22
127 Y=10
128 GOSUB 140
129 X1=7
130 X2=19
131 Y=13
132 GOSUB 140
133 X1=4
134 X2=16
135 Y=16
136 GOSUB 140
137 IF IN=1 THEN 153
138 IF VE=1 THEN 153
139 GOTO 154
140 FOR I=X1 TO X2 STE
    P 3
141 IF VE=1 THEN 148
142 CALL HCHAR(Y,I,CC)
143 CALL HCHAR(Y,I+1,C
    C+3)
144 CALL HCHAR(Y,I+2,C
    C+4)
145 CALL HCHAR(Y-1,I,C
    C+1)
146 CALL HCHAR(Y-1,I+1
    ,CC+2)
147 GOTO 150
148 CALL GCHAR(Y,I,A)
149 IF A<>CC THEN 420
150 NEXT I
151 IF VE=0 THEN 153
152 CALL SOUND(10,3000
    ,5)
153 RETURN
154 GOSUB 402
155 NP=INT((20-10+1)*R
    ND)+10
156 FOR P=1 TO NP
157 Y=0
158 Y=INT((16-4+1)*RND
    )+4
159 IF Y=4 THEN 165
160 IF Y=7 THEN 173
161 IF Y=10 THEN 181
162 IF Y=13 THEN 189
163 IF Y=16 THEN 197
164 GOTO 157
165 X=0
166 X=INT((28-16+1)*RN
    D)+16
167 IF X=16 THEN 212
168 IF X=19 THEN 212
169 IF X=22 THEN 212
170 IF X=25 THEN 212
171 IF X=28 THEN 212
```




```
172 GOTO 165
173 X=0
174 X=INT((25-13+1)*RN
    D)+13
175 IF X=13 THEN 212
176 IF X=16 THEN 212
177 IF X=19 THEN 212
178 IF X=22 THEN 212
179 IF X=25 THEN 212
180 GOTO 173
181 X=0
182 X=INT((22-10+1)*RN
    D)+10
183 IF X=10 THEN 212
184 IF X=13 THEN 212
185 IF X=16 THEN 212
186 IF X=19 THEN 212
187 IF X=22 THEN 212
188 GOTO 181
189 X=0
```

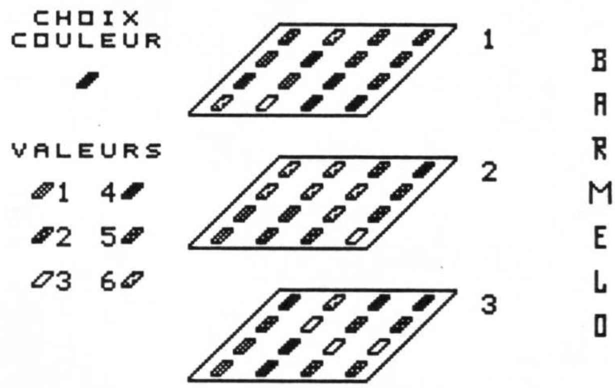
```
190 X=INT((19-7+1)*RND
    )+7
191 IF X=7 THEN 212
192 IF X=10 THEN 212
193 IF X=13 THEN 212
194 IF X=16 THEN 212
195 IF X=19 THEN 212
196 GOTO 189
197 X=0
198 X=INT((16-4+1)*RND
    )+4
199 IF X=4 THEN 212
200 IF X=7 THEN 212
201 IF X=10 THEN 212
202 IF X=13 THEN 212
203 IF X=16 THEN 212
204 GOTO 197
205 NEXT P
206 P=21
207 GOSUB 477
208 GOSUB 411
209 GOSUB 477
210 Y=10
211 X=16
212 CALL GCHAR(Y,X,A)
213 IF A>33 THEN 218
214 A=33
215 B=80
216 C=12
217 GOTO 236
218 IF A>40 THEN 223
219 A=40
220 B=33
221 C=6
222 GOTO 236
223 IF A>64 THEN 228
224 A=64
225 B=40
226 C=9
227 GOTO 236
228 IF A>72 THEN 233
229 A=72
230 B=64
231 C=13
232 GOTO 236
233 A=80
234 B=72
235 C=16
236 IF P<=NP THEN 289
237 CALL COLOR(8,2,C)
238 CALL HCHAR(Y,X,88)
239 CALL HCHAR(Y,X,A)
240 CALL KEY(3,T,S)
241 IF S=0 THEN 238
242 IF T<>86 THEN 246
243 VE=1
244 GOSUB 117
245 GOTO 383
246 IF T<>73 THEN 250
247 IN=1
248 GOSUB 117
249 GOTO 451
250 IF T<>68 THEN 263
251 IF Y<>4 THEN 253
252 IF X=28 THEN 238
253 IF Y<>7 THEN 255
254 IF X=25 THEN 238
255 IF Y<>10 THEN 257
256 IF X=22 THEN 238
257 IF Y<>13 THEN 259
258 IF X=19 THEN 238
259 IF Y<>16 THEN 261
260 IF X=16 THEN 238
261 X=X+3
262 GOTO 212
263 IF T<>69 THEN 268
264 IF Y=4 THEN 238
265 Y=Y-3
266 X=X+3
267 GOTO 212
268 IF T<>88 THEN 273
269 IF Y=16 THEN 238
270 Y=Y+3
271 X=X-3
272 GOTO 212
273 IF T<>83 THEN 286
274 IF Y>4 THEN 276
275 IF X=16 THEN 238
276 IF Y>7 THEN 278
277 IF X=13 THEN 238
278 IF Y>10 THEN 280
279 IF X=10 THEN 238
280 IF Y>13 THEN 282
281 IF X=7 THEN 238
282 IF Y<>16 THEN 284
283 IF X=4 THEN 238
284 X=X-3
285 GOTO 212
286 IF T<>13 THEN 240
287 NJ=NJ+1
288 IF NJ>150 THEN 442
289 CALL SOUND(-130,-4
    ,0,4000,10,40000,1
    0,40000,10)
290 CALL HCHAR(Y,X,B)
291 CALL HCHAR(Y,X+1,B
    +3)
292 CALL HCHAR(Y,X+2,B
    +4)
293 CALL HCHAR(Y-1,X,B
    +1)
294 CALL HCHAR(Y-1,X+1
    ,B+2)
295 IF P<>21 THEN 313
296 CU=CU+1
297 IF CU>57 THEN 300
298 CALL HCHAR(1,28,CU
    )
299 GOTO 313
300 CU=48
301 CD=CD+1
302 IF CD>57 THEN 306
303 CALL HCHAR(1,27,CD
    )
304 CALL HCHAR(1,28,CU
    )
305 GOTO 313
306 CE=CE+1
307 IF CE>57 THEN 442
308 CD=48
309 CU=48
310 CALL HCHAR(1,28,CU
    )
311 CALL HCHAR(1,27,CD
    )
312 CALL HCHAR(1,26,CE
    )
313 WA=Y
314 ZA=X
315 Y=0
316 X=0
317 IF WA=4 THEN 321
318 W=WA-3
319 Z=ZA+3
320 GOSUB 359
321 IF WA<>16 THEN 323
322 IF ZA=16 THEN 334
323 IF WA<>13 THEN 325
324 IF ZA=19 THEN 334
325 IF WA<>10 THEN 327
326 IF ZA=22 THEN 334
327 IF WA<>7 THEN 329
328 IF ZA=25 THEN 334
329 IF WA<>4 THEN 331
330 IF ZA=28 THEN 334
331 W=WA
332 Z=ZA+3
333 GOSUB 359
334 IF WA=16 THEN 338
335 W=WA+3
336 Z=ZA-3
337 GOSUB 359
338 IF WA<>16 THEN 340
339 IF ZA=4 THEN 351
340 IF WA<>13 THEN 342
341 IF ZA=7 THEN 351
342 IF WA<>10 THEN 344
343 IF ZA=10 THEN 351
344 IF WA<>7 THEN 346
345 IF ZA=13 THEN 351
346 IF WA<>4 THEN 348
347 IF ZA=16 THEN 351
348 W=WA
349 Z=ZA-3
350 GOSUB 359
351 Y=WA
352 X=ZA
353 WA=0
354 ZA=0
355 B=0
356 C=0
357 IF P=21 THEN 212
358 IF P<NP THEN 205 E
    LSE 206
359 CALL GCHAR(W,Z,A)
360 IF A>33 THEN 363
```

```
361 A=40
362 GOTO 373
363 IF A>40 THEN 366
364 A=64
365 GOTO 373
366 IF A>64 THEN 369
367 A=72
368 GOTO 373
369 IF A>72 THEN 372
370 A=80
371 GOTO 373
372 A=33
373 CALL SOUND(100,110
    ,15,-4,15,110,15)
374 CALL HCHAR(W,Z,A)
375 CALL HCHAR(W,Z+1,A
    +3)
376 CALL HCHAR(W,Z+2,A
    +4)
377 CALL HCHAR(W-1,Z,A
    +1)
378 CALL HCHAR(W-1,Z+1
    ,A+2)
379 A=0
380 W=0
381 Z=0
382 RETURN
383 N=523
384 FOR I=1 TO 15
385 CALL SOUND(100,N,5
    )
386 N=N+110
387 NEXT I
388 GOTO 435
389 M$="je"
390 U=1
391 V=4
392 GOSUB 465
393 M$="choisi"
394 U=2
395 V=2
396 GOSUB 470
397 M$="le"
398 U=3
399 V=4
400 GOSUB 470
401 RETURN
402 M$="brouillage"
403 U=21
404 V=5
405 GOSUB 465
406 M$="du jeu"
407 U=23
408 V=7
409 GOSUB 470
410 RETURN
411 M$="termine"
412 U=21
413 V=6
414 GOSUB 465
415 M$="a vous de joue
    r"
416 U=23
417 V=2
418 GOSUB 470
419 RETURN
420 M$="incomplet"
421 U=20
422 V=5
423 GOSUB 465
424 M$="continuez"
425 U=22
426 V=5
427 GOSUB 470
428 M$="la partie"
429 U=24
430 V=5
431 GOSUB 470
432 VE=0
433 GOSUB 477
434 GOTO 210
435 M$="partie gagnee"
436 U=22
437 V=3
438 PG=1
439 VE=0
440 GOSUB 465
441 GOTO 455
442 M$="score  maximum
    "
443 U=20
444 V=3
445 GOSUB 465
446 M$="atteint"
447 U=22
448 V=5
449 GOSUB 470
450 GOTO 499
451 M$="partie nulle"
452 U=22
453 V=3
454 GOSUB 465
455 GOSUB 477
456 M$="nouvelle parti
    e"
457 U=21
458 V=2
459 GOSUB 470
460 M$="o    n"
461 U=23
462 V=7
463 GOSUB 470
464 GOTO 481
465 FOR I=1 TO 5
466 CALL SCREEN(4)
467 CALL SOUND(10,-1,1
    0)
468 CALL SCREEN(2)
469 NEXT I
470 FOR I=1 TO LEN(M$)
471 CALL HCHAR(U,V+I,A
    SC(SEG$(M$,I,1)))
472 NEXT I
473 U=0
474 V=0
475 M$=""
476 RETURN
477 FOR I=3 TO 17
478 CALL VCHAR(20,I,32
    ,5)
479 NEXT I
480 RETURN
481 CALL KEY(3,T,S)
482 IF S=0 THEN 481
483 IF T<>79 THEN 498
484 IF IN=1 THEN 488
485 FOR I=3 TO 8
486 CALL VCHAR(1,I,32,
    6)
487 NEXT I
488 GOSUB 477
489 CALL HCHAR(1,26,48
    ,3)
490 CU=48
491 CD=48
492 CE=48
493 NP=0
494 NJ=0
495 P=0
496 IF IN=1 THEN 154
497 GOTO 89
498 IF T<>78 THEN 481
499 CALL CLEAR
500 IF IN=1 THEN 504
501 IF PG=1 THEN 506
502 PRINT "    vous ave
    z tout essaye"::"
    vous ferez peut et
    re mieux"::"      u
    ne prochaine fois"
    :::
503 GOTO 508
504 PRINT "alors comme
     ca on abandonne":
    :" ce n est pas ch
    ic pour moi"::"  j
    e commencais a m a
    muser":::
505 GOTO 508
506 PRINT " que c est
    chouette d avoir":
    :"  un partenaire
    tres doue"::"j es
    pere jouer souvent
     avec"::
507 PRINT "
     vous":::
508 FOR I=1 TO 1000
509 NEXT I
510 CALL CLEAR
511 PRINT " vous avez
    joue ";NJ;" fois":
    ::::
512 PRINT "    vous dev
    iez reussir en"::T
    AB(10);NP*5;" coup
```

```
     s"::::TAB(11);"au
      revoir"
513 GOSUB 523
514 END
515 DATA 1,349,294,147
    ,1,440,349,175,2,5
    23,392,330,8,698,5
    87,117,2,466,233,1
    17,1,349,294,147,1
    ,587,294,147
516 DATA 8,523,440,131
    ,2,440,349,175,2,3
    49,294,147,1,262,1
    96,165,1,466,233,1
    17,8,392,330,165,2
    ,392,330,165
517 DATA 2,349,294,147
    ,2,392,330,165,8,4
    40,349,175,2,392,2
    33,131,2,349,233,1
    31,2,330,233,131,8
    ,349,220,175
518 DATA 2,523,262,131
    ,2,392,196,131,1,4
    40,220,131,1,466,2
    33,131,2,523,440,1
    75,1,523,440,175,8
    ,523,440,175
519 DATA 2,523,262,131
    ,2,392,196,131,1,4
    40,220,131,1,466,2
    33,131,6,523,440,1
    75,2,523,440,175,2
    ,466,233,175
520 DATA 1,523,392,330
    ,1,587,294,147,2,6
    59,523,440,2,659,5
    23,440,8,659,523,4
    40,2,659,330,165,2
    ,466,233,117
521 DATA 1,523,262,131
    ,1,587,294,147,8,6
    59,466,131,2,392,3
    30,165,2,392,233,1
    31,1,349,294,147,4
    ,330,165,131
522 DATA 6,349,220,175
    ,0,0,0,0
523 RESTORE 515
524 READ D,N1,N2,N3
525 IF D=0 THEN 532
526 IF IN=0 THEN 527 E
    LSE 528
527 IF PG=0 THEN 530
528 CALL SOUND(275*D,N
    1,10,N2,12,N3,15)
529 GOTO 524
530 CALL SOUND(-40*D,N
    1,15,N2,15,N3,12)
531 GOTO 524
532 RETURN
```

Barmélo 2

```
10 REM **************
11 REM *            *
12 REM *  BARMELO 2 *
13 REM *            *
14 REM *  BASIC  TI *
15 REM *            *
16 REM *  Copyright *
17 REM *            *
18 REM * 99 Magazine *
19 REM *            *
20 REM * G.  Goument *
21 REM *            *
22 REM **************
23 CALL CLEAR
24 CALL SCREEN(16)
25 RANDOMIZE
26 A=0
27 B=0
28 D=0
29 P=0
30 C=0
31 VE=0
32 NP=0
33 J=0
34 AN=0
35 CC=0
36 Y=0
37 YA=0
38 X=0
39 XA=0
40 U=0
41 V=0
42 GOSUB 525
43 FOR I=1 TO 8
44 READ Z,W$
45 CALL CHAR(Z,W$)
46 NEXT I
47 DATA 41,7C242438242
   4247C,55,81C3A59981
   818181,57,3C24243C2
   4242424,65,7C24243C
   30282424
48 DATA 73,3C202038202
   0203C,81,2020202020
   24243C,89,3C2424242
   424243C,128,FFFFFFF
   FFFFFFFF
49 PRINT " georges"::"
     goument":::TAB(
   8);"vous  propose":
   :::::::::TAB(6);"j
   eu de  reflexion"::
   ::
50 PRINT "          niveau
    champion":::
51 X=12
52 FOR I=1 TO 7
53 READ A
54 CALL HCHAR(12,X,A)
55 X=X+1
56 A=0
57 NEXT I
58 DATA 41,57,65,55,73
   ,81,89
59 CALL HCHAR(12,20,50
   )
60 FOR I=9 TO 12
61 CALL COLOR(I,7,1)
62 NEXT I
63 CALL SOUND(300,523,
   15)
64 CALL COLOR(2,13,1)
65 CALL COLOR(4,9,1)
66 CALL COLOR(5,5,1)
67 CALL COLOR(3,8,1)
68 CALL COLOR(6,10,1)
69 CALL COLOR(7,14,1)
70 CALL COLOR(8,3,1)
```




```
71 GOSUB 548
72 CALL CLEAR
73 GOSUB 541
74 M$="patientez quelq
   ues secondes"
75 X=2
76 GOSUB 529
77 CALL CLEAR
78 CALL SCREEN(2)
79 GOSUB 525
80 CALL CHAR(35,"00000
   000000000FF")
81 CALL CHAR(36,"FF")
82 CALL CHAR(37,"01020
   4081020408")
83 RESTORE 88
84 FOR I=1 TO 6
85 READ O
86 CALL CHAR(O,"000F1F
   3E7CF8F")
87 NEXT I
88 DATA 40,56,64,72,80
   ,88
89 PRINT " choix
   ########":"couleur
       %        % 1":"
             %        %
          )":"          %
            %"
```

```
90 PRINT "        %
       %         9":"
            $$$$$$$$":"v
   aleurs      ########
           A"
91 PRINT "              %
            % 2":" (1 4H
       %     %
    7":"          %
      %"
92 PRINT " 82 5P  %
       %         I":"
            $$$$$$$$":"
   @3 6X       ########
           Q"
93 PRINT "              %
            % 3":"
       %     %
    Y":"          %
      %"
94 PRINT "        %
       %":"          $$$
   $$$$$":::::
95 FOR I=1 TO 12
96 CALL COLOR(I,8,1)
97 NEXT I
98 CALL COLOR(2,13,1)
99 CALL COLOR(4,9,1)
100 CALL COLOR(5,15,1)
101 CALL COLOR(6,14,1)
102 CALL COLOR(7,12,1)
103 CALL COLOR(8,6,1)
104 GOSUB 541
105 CC=0
106 CC=INT((88-40+1)*R
    ND)+40
107 CALL SOUND(8,1984,
    12)
108 IF CC=40 THEN 115
109 IF CC=56 THEN 115
110 IF CC=64 THEN 115
111 IF CC=72 THEN 115
112 IF CC=80 THEN 115
113 IF CC=88 THEN 115
114 GOTO 105
115 GOSUB 117
116 GOTO 140
117 CALL HCHAR(4,6,CC)
118 Y=2
119 FOR I=1 TO 3
120 IF VE=0 THEN 122
121 CALL SOUND(10,3500
    ,10)
122 X=15
123 FOR K=1 TO 4
124 FOR L=1 TO 4
125 IF VE=0 THEN 129
126 CALL GCHAR(Y,X,A)
127 IF A<>CC THEN 520
128 GOTO 130
129 CALL HCHAR(Y,X,CC)
130 X=X+2
131 NEXT L
132 IF VE=0 THEN 134
133 CALL SOUND(12,1966
    ,20)
134 Y=Y+1
135 X=X-9
136 NEXT K
137 Y=Y+2
138 NEXT I
139 RETURN
140 M$="brouillage des
     barres"
141 X=5
142 GOSUB 529
143 NP=INT((30-15+1)*R
    ND)+15
144 FOR P=1 TO NP
145 J=0
146 X=0
147 Y=0
148 J=INT((30-1+1)*RND
    )+1
149 IF J>20 THEN 163
150 IF J>10 THEN 157
151 Y=INT((5-2+1)*RND)
    +2
152 IF Y=2 THEN 169
153 IF Y=3 THEN 175
154 IF Y=4 THEN 181
155 IF Y=5 THEN 187
156 GOTO 151
157 Y=INT((11-8+1)*RND
    )+8
158 IF Y=8 THEN 169
159 IF Y=9 THEN 175
160 IF Y=10 THEN 181
161 IF Y=11 THEN 187
162 GOTO 157
163 Y=INT((17-14+1)*RN
    D)+14
164 IF Y=14 THEN 169
165 IF Y=15 THEN 175
166 IF Y=16 THEN 181
167 IF Y=17 THEN 187
168 GOTO 163
169 X=INT((21-15+1)*RN
    D)+15
170 IF X=15 THEN 202
171 IF X=17 THEN 202
172 IF X=19 THEN 202
173 IF X=21 THEN 202
174 GOTO 169
175 X=INT((20-14+1)*RN
    D)+14
176 IF X=14 THEN 202
177 IF X=16 THEN 202
178 IF X=18 THEN 202
179 IF X=20 THEN 202
180 GOTO 175
181 X=INT((19-13+1)*RN
    D)+13
182 IF X=13 THEN 202
183 IF X=15 THEN 202
184 IF X=17 THEN 202
185 IF X=19 THEN 202
186 GOTO 181
187 X=INT((18-12+1)*RN
    D)+12
188 IF X=12 THEN 202
189 IF X=14 THEN 202
190 IF X=16 THEN 202
191 IF X=18 THEN 202
192 GOTO 187
193 NEXT P
194 P=31
195 CJ=0
196 GOSUB 541
197 M$="termine   a vo
    us de jouer"
198 X=3
199 GOSUB 529
200 GOSUB 541
201 GOTO 229
202 CALL GCHAR(Y,X,A)
203 IF A>40 THEN 207
204 A=40
205 B=88
206 GOTO 225
207 IF A>56 THEN 211
208 A=56
209 B=40
210 GOTO 225
211 IF A>64 THEN 215
212 A=64
213 B=56
214 GOTO 225
215 IF A>72 THEN 219
216 A=72
217 B=64
218 GOTO 225
219 IF A>80 THEN 223
220 A=80
221 B=72
222 GOTO 225
223 A=88
224 B=80
225 IF P<=NP THEN 349
226 IF J=0 THEN 229
227 CALL HCHAR(Y,X,32)
228 CALL HCHAR(Y,X,A)
229 CALL KEY(3,T,S)
230 IF S=0 THEN 226
231 IF T<>73 THEN 234
232 AN=1
233 GOTO 494
234 IF T<>86 THEN 240
235 VE=1
236 J=0
237 GOSUB 118
238 PG=1
239 GOTO 499
240 IF T<>49 THEN 244
241 J=1
242 Y=3
```

```
243 GOTO 251
244 IF T<>50 THEN 248
245 J=2
246 Y=9
247 GOTO 251
248 IF T<>51 THEN 254
249 J=3
250 Y=15
251 X=16
252 CALL SOUND(20,1966
    ,12)
253 GOTO 202
254 IF T<>69 THEN 267
255 IF J=0 THEN 229
256 IF J>1 THEN 259
257 IF Y=2 THEN 227
258 GOTO 263
259 IF J>2 THEN 262
260 IF Y=8 THEN 227
261 GOTO 263
262 IF Y=14 THEN 227
263 GOSUB 299
264 Y=YA-1
265 X=XA+1
266 GOTO 294
267 IF T<>88 THEN 280
268 IF J=0 THEN 229
269 IF J>1 THEN 272
270 IF Y=5 THEN 227
271 GOTO 276
272 IF J>2 THEN 275
273 IF Y=11 THEN 227
274 GOTO 276
275 IF Y=17 THEN 227
276 GOSUB 299
277 Y=YA+1
278 X=XA-1
279 GOTO 294
280 IF T<>68 THEN 287
281 IF J=0 THEN 229
282 G=0
283 GOSUB 304
284 GOSUB 299
285 X=XA+2
286 GOTO 293
287 IF T<>83 THEN 347
288 IF J=0 THEN 229
289 G=1
290 GOSUB 304
291 GOSUB 299
292 X=XA-2
293 Y=YA
294 CALL HCHAR(YA,XA,A
    )
295 YA=0
296 XA=0
297 CALL SOUND(5,1940,
    18)
298 GOTO 202
299 YA=Y
300 XA=X
301 Y=0
302 X=0
303 RETURN
304 IF J>1 THEN 328
305 IF Y>2 THEN 311
306 IF G=1 THEN 309
307 IF X=21 THEN 227
308 RETURN
309 IF X=15 THEN 227
310 RETURN
311 IF Y>3 THEN 317
312 IF G=1 THEN 315
313 IF X=20 THEN 227
314 RETURN
315 IF X=14 THEN 227
316 RETURN
317 IF Y>4 THEN 323
318 IF G=1 THEN 321
319 IF X=19 THEN 227
320 RETURN
321 IF X=13 THEN 227
322 RETURN
323 IF G=1 THEN 326
324 IF X=18 THEN 227
325 RETURN
326 IF X=12 THEN 227
327 RETURN
328 IF J>2 THEN 338
329 IF Y>8 THEN 332
330 IF G=1 THEN 309
331 GOTO 307
332 IF Y>9 THEN 335
333 IF G=1 THEN 315
334 GOTO 313
335 IF Y>10 THEN 323
336 IF G=1 THEN 321
337 GOTO 319
338 IF Y>14 THEN 341
339 IF G=1 THEN 309
340 GOTO 307
341 IF Y>15 THEN 344
342 IF G=1 THEN 315
343 GOTO 313
344 IF Y>16 THEN 323
345 IF G=1 THEN 321
346 GOTO 319
347 IF T<>13 THEN 229
348 IF J=0 THEN 229
349 CALL SOUND(120,110
    ,15,-4,15,110,15)
350 CALL HCHAR(Y,X,B)
351 IF P<31 THEN 354
352 CJ=CJ+1
353 IF CJ>200 THEN 510
354 U=Y
355 V=X
356 Y=0
357 X=0
358 IF U=2 THEN 362
359 IF U=8 THEN 362
360 IF U=14 THEN 362
361 GOTO 365
362 IF V=15 THEN 392
363 IF V=21 THEN 396
364 GOTO 408
365 IF U=5 THEN 369
366 IF U=11 THEN 369
367 IF U=17 THEN 369
368 GOTO 372
369 IF V=12 THEN 400
370 IF V=18 THEN 404
371 GOTO 413
372 IF V=13 THEN 382
373 IF V=14 THEN 382
374 IF V=19 THEN 387
375 IF V=20 THEN 387
376 GOSUB 428
377 GOSUB 431
378 GOSUB 434
379 GOSUB 437
380 GOSUB 440
381 GOTO 417
382 GOSUB 428
383 GOSUB 431
384 GOSUB 434
385 GOSUB 440
386 GOTO 417
387 GOSUB 428
388 GOSUB 437
389 GOSUB 434
390 GOSUB 440
391 GOTO 417
392 GOSUB 431
393 GOSUB 434
394 GOSUB 440
395 GOTO 417
396 GOSUB 437
397 GOSUB 434
398 GOSUB 440
399 GOTO 417
400 GOSUB 428
401 GOSUB 431
402 GOSUB 440
403 GOTO 417
404 GOSUB 428
405 GOSUB 437
406 GOSUB 440
407 GOTO 417
408 GOSUB 431
409 GOSUB 434
410 GOSUB 437
411 GOSUB 440
412 GOTO 417
413 GOSUB 437
414 GOSUB 428
415 GOSUB 431
416 GOSUB 440
417 U=Y
418 V=X
419 U=0
420 V=0
421 J=0
422 JY=0
423 A=0
424 B=0
```

```
425 C=0
426 IF P=31 THEN 229
427 IF P<NP THEN 193 E
    LSE 194
428 CALL GCHAR(U-1,V+1
    ,A)
429 C=1
430 GOTO 447
431 CALL GCHAR(U,V+2,A
    )
432 C=2
433 GOTO 447
434 CALL GCHAR(U+1,V-1
    ,A)
435 C=3
436 GOTO 447
437 CALL GCHAR(U,V-2,A
    )
438 C=4
439 GOTO 447
440 IF U>11 THEN 444
441 CALL GCHAR(U+6,V,A
    )
442 C=5
443 GOTO 447
444 IF U<7 THEN 481
445 CALL GCHAR(U-6,V,A
    )
446 C=6
447 IF A>40 THEN 450
448 A=56
449 GOTO 463
450 IF A>56 THEN 453
451 A=64
452 GOTO 463
453 IF A>64 THEN 456
454 A=72
455 GOTO 463
456 IF A>72 THEN 459
457 A=80
458 GOTO 463
459 IF A>80 THEN 462
460 A=88
461 GOTO 463
462 A=40
463 CALL SOUND(12,1967
    ,20)
464 IF C>1 THEN 467
465 CALL HCHAR(U-1,V+1
    ,A)
466 GOTO 480
467 IF C>2 THEN 470
468 CALL HCHAR(U,V+2,A
    )
469 GOTO 480
470 IF C>3 THEN 473
471 CALL HCHAR(U+1,V-1
    ,A)
472 GOTO 480
473 IF C>4 THEN 476
474 CALL HCHAR(U,V-2,A
    )
475 GOTO 480
476 IF C>5 THEN 479
477 CALL HCHAR(U+6,V,A
    )
478 GOTO 480
479 CALL HCHAR(U-6,V,A
    )
480 IF C=5 THEN 444
481 RETURN
482 M$="voulez vous re
    jouer   o  n"
483 X=3
484 GOSUB 529
485 CALL KEY(3,T,S)
486 IF S=0 THEN 485
487 IF T<>79 THEN 498
488 GOSUB 541
489 CC=32
490 AN=0
491 CJ=0
492 GOSUB 117
493 GOTO 105
494 M$="annulation de
    la partie"
495 X=4
496 GOSUB 529
497 GOTO 499
498 IF T<>78 THEN 485
499 CALL CLEAR
500 IF AN=0 THEN 506
501 CALL SCREEN(9)
502 PRINT "  alors les
     nerfs craquent"::
    "vous voyez  quand
     je le veux"::" pe
    rsonne ne peut me
    battre":::::
503 PRINT "          a
    bientot"::::::::
504 GOSUB 548
505 GOTO 510
506 CALL SCREEN(13)
507 PRINT "    la part
    ie est gagnee":::"
        vous etes form
    idable"::"  j espe
    re rejouer souvent
    "
508 PRINT "          av
    ec vous"::::
509 GOTO 503
510 CALL CLEAR
511 NJ=NP*6
512 PRINT "  j ai brou
    ille ";NP;"fois"::
    " l  ensemble des
    3 plateaux":::
513 PRINT " vous devie
    z reussir le jeu":
    :"      en ";NJ;"
    coups":::
514 PRINT "       vous a
    vez effectue"::TAB
    (7);CJ;"permutatio
    ns":::::
515 IF AN=0 THEN 518
516 PRINT "    et tout
    cela pour rien"
517 GOTO 519
518 PRINT "vous vous e
    n etes bien sorti"
519 END
520 M$="incomplet cont
    inuez le jeu"
521 X=3
522 GOSUB 529
523 GOSUB 541
524 GOTO 229
525 FOR I=1 TO 12
526 CALL COLOR(I,1,1)
527 NEXT I
528 RETURN
529 FOR I=1 TO 5
530 CALL COLOR(13,8,1)
531 CALL SOUND(5,-1,15
    )
532 CALL COLOR(13,1,1)
533 NEXT I
534 FOR I=1 TO LEN(M$)
535 CALL HCHAR(24,X+I,
    ASC(SEG$(M$,I,1)))
536 NEXT I
537 FOR I=1 TO 100
538 NEXT I
539 X=0
540 RETURN
541 CALL COLOR(13,1,1)
542 CALL HCHAR(24,1,12
    8,32)
543 RETURN
544 DATA 2,784,2,880,4
    ,988,6,1047,2,988,
    2,880,2,784,2,880,
    4,784,6,659,2,523,
    2,587,2,659,2,587
545 DATA 4,523,6,659,2
    ,784,2,820,2,988,4
    ,880,6,784,2,659,2
    ,659,4,659,6,831,2
    ,659,2,740,2,831
546 DATA 2,880,4,988,6
    ,1047,2,880,4,988,
    6,1047,2,988,4,880
    ,6,1175,2,988,4,78
    4,6,1047,2,880
547 DATA 2,740,4,831,6
    ,831,8,880,0,0
548 RESTORE 544
549 READ D,N
550 IF D=0 THEN 553
551 CALL SOUND(D*-50,N
    *4,10,N*2,12,N/4,1
    4)
552 GOTO 549
553 RETURN
```

La disquette 'Fichier d'adresses' Texas Instruments PHD 5001/FR

J-B. Morelle

N'étant pas un spécialiste de la programmation, les renseignements que je vais donner ci-dessous concerneront uniquement le problème de la saisie de ce programme.

D'autre part comme il existe chez Texas un module "Gestion de Fichiers" bien connu par de nombreux Texans, il m'a semblé intéressant de comparer leurs avantages et inconvénients respectifs.

La disquette "Fichier d'adresses" est en vente à la FNAC Wagram et à la Règle à Calcul à Paris au prix de 415 F.

Il est curieux de constater le manque de protection de ce logiciel qui peut être copié sans problème. Il n'est évidemment pas question de vous communiquer le listing de son contenu ne tenant pas à avoir, pas plus que 99 magazine, des ennuis avec la maison Texas...

Contenu de la disquette

La disquette contient 7 programmes :

- Initifich
- Fichieradr
- Impfichier
- Condfich
- trifichier
- recherfich
- Contenu

Programme Initifich

Il est nécessaire tout d'abord de formater la disquette à l'aide du module Disk Manager mais ce n'est pas suffisant ! Il faut en effet initialiser la disquette vierge à l'aide du programme Initifich et elle ne pourra rien contenir d'autre que le fichier créé. En effet, si on fait le catalogue de cette disquette, la place libre est nulle.

Programme fichieradr

Pour appeler ce programme, il faut auparavant donner le fameux coup de balai : Call files (1), New, et Old...
Apparait alors le menu suivant :

1 - Recherche
2 - Mise à jour
3 - Noms de groupe
4 - Sortie

Il y a 13 rubriques par personne plus 10 groupes dans lesquels on peut la faire entrer.

1 - Recherche

Trois options s'affichent :

1 - une personne
2 - par groupe
3 - toute personne fichée

Il est nécessaire d'entrer le nom et le prénom exacts sinon le programme vous signale qu'il ne connait pas la personne.
Sinon, vous pouvez soit prendre l'option 3 et toutes les personnes fichées défilent une par une, contrôlées par Enter. Il est certain que si la personne cherchée est à la 270ème place, ce sera assez long. La solution consiste à utiliser le programme *Recherfich* qui sert uniquement à trouver un nom en ne connaissant que les 3 premières lettres.

2 - Mise à jour

Trois options s'affichent :

1 - Ajouter un nom
2 - Mise à jour d'un nom
3 - Effacer un nom

Un seul commentaire : le nom effacé reste dans le fichier (en cas d'erreur), et ne sera supprimé que si vous faites tourner le programme condfich. Vous pressentez déjà la lourdeur du système !

Si vous choisissez l'optiion 2, mise à jour d'un nom, on vous demande le numéro de la ligne à

changer correspondant à la rubrique qui vous intéresse. Il y a 8 rubriques, 3 commentaires et 10 groupes possibles.

Programme Condefich

Trois options apparaissent :

1 - supprimer les noms marqués effacement
2 - suppprimer les noms marqués effacement ou inactifs.

Là, il y a un risque car les inactifs sont ceux n'appartenant à aucun groupe. C'est à dire que si vos personnes ne sont dans aucun groupe et que vous appuyez sur 2 par erreur, tout votre fichier est effacé. L'opération de Condefich est longue. Ce programme rétablit en même temps l'ordre alphabétique.

Programme Trifichier

Il correspond à l'option Analyse de fiches du module Gestion de Fichiers. Il permet de faire un tri suivant 11 critères et d'imprimer les éléments choisis dans l'ordre désiré.

Programme Impfichier

Là encore, il est nécessaire de taper en premier Call Files(1), etc... Sont demandées ensuite les références de l'unité d'impression (par exemple : RS232.BA=1200 s'il s'agit d'une imprimante série fonctionnant à 1200 bauds).

Quatre options sont affichées :

1 - Toutes informations sur le fichier
2 - Nom, adresse, téléphone
3 - Etiquette
4 - Sortie

La première option imprime tous les renseignements, mais les uns au-dessous des autres. Pour avoir un tableau il faut choisir un nombre réduit de rubriques car plusieurs colonnes sont prévues pour chacune d'elles et elles ne tiennent pas sur les 80 colonnes prévues à l'impression.
La deuxième option agit de même mais d'indique que les noms, adresses et téléphone.
Pour la troisième, les étiquettes, il vous est demandé leur format et le nombre par rangée. Elles sont encadrées par un petit dessin.

"Fichier d'Adresses" : des avantages...

Chaque rubrique contient beaucoup plus de place pour écrire les informations. Il est possible de stocker 350 fiches quelle que soit la longueur des renseignements saisis, ce qui est appréciable. L'extension 32 Ko ne semble pas nécessaire puisque la mienne n'est jamais allumée. La grande rapidité d'impression est très utile pour le traitement de longs fichiers.

...mais aussi des inconvénients

Très grande lourdeur dans la manipulation des programmes. Le possesseur d'un seule unité de disquette risque fort de se tromper. Si vous utiliser 2 lecteurs, il faut mettre le disque de données dans le lecteur 1 et le disque programme dans le 2 ; Ainsi, vous n'aurez aucune manipulation à faire.
Au niveau de la saisie des données, le nom de la ville et celui du code postal sont systématiquement liés ce qui peut-être gênant si vous utilisez ce programme pour la gestion de votre cave à vins...

Avantages du module "Gestion de Fichiers"

Très grande souplesse pour passer de la lecture à la modification ou à l'impression des fiches. L'impression se fait sur un tableau qui utilise toute la largeur de la page. Il est possible de faire des tris à l'impression d'un fa\on très pratique. Par exemple pour ressortir les juniors d'un club de tennis si vous avez saisi tous les membres du club.

Inconvénients du module "Gestion de fichiers"

Il n'y a que peu de place par ligne pour entrer les données. Quinze rubriques seulement, qui, si elles sont toutes utilisées réduisent considérablement le nombres de fiches disponibles. Par exemple, si vous entrez nom, prénom, date de naissance, ville, classement, vous allez plafonner à quatre vingt dix fiches, ce qui est peu pour une association.
Il est très lent à l'impression : vous pouvez mettre l'imprimante en route et aller faire un tour (près de trente minutes pour cent soixante-deux fiches avec une Mannesman Tally 132 colonnes 9600 bauds...)
Un reclassement alphabétique peut nécessiter vingt à trente minutes.

Conclusion

Si vous faites de petits fichiers, utilisez le module Gestion de Fichiers. Dans le cas contraire, la disquette Fichier d'Adresses semble indiquée.

99

Des caractères dans la page graphique haute résolution

Gérard Baroni
et
Jean Marin

Dans les précédents numéros de "99 Magazine", nous vous avons décrit la manière de faire coexister pacifiquement le mode Bitmap et le Basic, en configuration console/Mini-mémoire/ lecteur de cassettes. Vous avez de même la possibilité de sauvegarder des fichiers d'images rapidement sur cassette.

Le propos du présent article est de vous permèttre de mélanger du texte avec du graphisme Bitmap. Jusqu'à présent, un problème se posait à ce sujet; vu le peu de place disponible en mémoire vive, les formes des lettres disparaissaient lors de l'initialisation du mode Bitmap, et il devenait impossible d'écrire.

Notre façon de résoudre ce problème consiste à aller chercher les formes des lettres là où elles se trouvent, c'est-à-dire en mémoire morte.

Un examen des 8Ko de la ROM, et des 18Ko des GROMs de la console (trois GROMs de 6Ko en console) nous montre que les formes des lettres sont en GROM. Dans la GROM, nous découvrons deux jeux de caractères:

1) les grandes lettres de la mire d'introduction. Codées sur huit octets par lettre, elles se trouvent en GROM de l'adresse >04B4 ou se trouve l'espace - code ASCII 32 - huit octets nuls, jusqu'en >06B3 où nous terminons ce jeu avec le code ASCII 95 (souligné). Vous remarquerez que ce jeu de caractères ne connait pas les minuscules.
2) les petites lettres utilisées normalement en Basic sont codées sur 7 octets seulement par lettre. Elles commencent en >06B4 (code ASCII 32 de l'espace) et se terminent en >094C (code ASCII 126 - tilde).

L'accès aux GROMs, qui est décrit dans le manuel Editeur/ assembleur, est plus difficile que l'accès à la ROM, parce que les GROMs, comme la RAM VDP, ne sont pas en prise directe avec le microprocesseur.

Pour le problème qui nous concerne, signalons un point important : c'est la nécessité de sauvegarder l'adresse courante de la GROM et de la restituer après accès à la ROM. En effet, notre intrusion détruit cette adresse courante et, comme le retour au Basic ne se fait correctement que si cette adresse n'a pas été modifiée, vous comprendrez l'intérêt de cette opération de sauvegarde. L'interpréteur Basic est situé dans les GROMs et il est indispensable de ne rien y perturber.

Le programme, dont le listing suit, vous permettra, en faisant un CALL LINK("TEXTE",A$, V,H,C) de mélanger du texte avec votre graphisme Bitmap.

A$ = chaîne à écrire, longueur maximum de 255 octets;
V = position verticale, de 0 à 23, du premier caractère de la chaîne;
H = position horizontale, de 0 à 31, du premier caractère de la chaîne;
C = 16 fois couleur des caractères plus couleur du fond (entre 0 et 255). Exemple : 240 = 15 * 16 + 0, ce qui donne des caractères blanc (15) sur fond transparent (0).

Quelques explications sur la routine "TEXTE"

En premier lieu, on commence à saisir les quatre paramètres A$, V, H et C. La chaîne A$ occupe un tampon commençant à l'adresse >7B00.

- le registre R0 est mis à zéro pour la saisie;
- le registre R1 contient le numéro du paramètre à saisir;

- le registre R2 pointe sur le tampon du texte;
- le registre R3 contient la longueur de A$;
- le registre R4 contient, à la fin des saisies, le numéro de la case d'écran 32 * V + H;
- le registre R6 contient, dans son octet haut, la couleur C;
- le registre R10 contient l'adresse de retour au Basic;
- le registre R11 contient l'adresse de retour du sous-programme de saisie.

Ensuite, on sauvegarde l'adresse courante GROM dans le registre R5. Après, suit une boucle de transfert des formes lues dans la GROM, en fonction du code ASCII, vers l'écran.
Au départ de la boucle se fait le test de sortie de cette boucle. La sortie a lieu soit s'il n'y a plus de caractères à afficher, soit si l'affichage demandé sort de l'écran.
Les formes de caractères sont obtenues en GROM à l'adresse >O3B4 + 8 * code ASCII.
Exemple : espace = 32 -> adresse >04B4 précédemment citée.
Remarquez qu'ici nous avons volontairement choisi les grands caractères de la mire d'introduction, ce qui suppose que le texte affiché ne contient pas de minuscules. Si vous désirez absolument les minuscules, nous vous laissons le soin d'adapter ce programme. Il vous faudra changer l'adresse de départ de la table en GROM, et le nombre d'octets par caractère, 7 au lieu de 8.

Dans la partie "Transfert GROM/ VDP", le registre R0 sert d'intermédiaire temporaire dans les calculs des adresses GROMS et VDP.
- R1 contient l'adresse de la table des formes en RAM VDP;
- R2 pointe sur le buffer "Texte";
- R3 contient le nombre d'octets restant à écrire;
- R4 pointe sur la table d'écran;
- R5 et R6 sont inchangés.

Une fois le transfert terminé, on restaure l'adresse GROM et on revient au Basic.

Avec cette dernière routine et celles étudiées ensembles dans les numéros précédents, nous pouvons maintenant disposer totalement du Bitmap à l'aide du Basic. Les 20Ko de RAM disponibles sont actuellement bien utilisés. Il nous reste encore de la place en Mini-mémoire pour écrire d'autres routines telles que peinture de surfaces, aide au dessin ou toute autre routine qu'il vous plaira de créer selon vos besoins.

Ne pas oublier la table REF/DEF: Nom de la routine "TEXTE" et adresse d'entrée >7C1C.

A Ticazur, actuellement, a débuté un cours de Jean Marin d'initiation au langage FORTH. Ce langage, évolué, nécessite la configuration suivante : Mini-mémoire ou Basic étendu, extension 32Ko et lecteur de cassettes.
De plus, Ticazur poursuit, sous l'impulsion de Robert Schenck, ses activités "hardware", débutées en fin d'année scolaire 1985.

99

Source de la routine

```
7AD0              AORG >7AD0
7AD0 04C0    E1   CLR  R0
7AD2 0581         INC  R1
7AD4 0420         BLWP @>6044
7AD6 6044
7AD8 0420         BLWP @>601C
7ADA 601C
7ADC 1200         DATA >1200
7ADE C020         MOV  @>834A,R0
7AE0 834A
7AE2 045B         B    *R11
7B00              AORG >7B00
7B00         BT   BSS  >FF
7C1C              AORG >7C1C
7C1C 04C0         CLR  R0
7C1E C28B         MOV  R11,R10
7C20 0201         LI   R1,>0001
7C22 0001
7C24 0202         LI   R2,BT
7C26 7B00
7C28 0712         SETO *R2
7C2A 0420         BLWP @>604C
7C2C 604C
7C2E 04C3         CLR  R3
7C30 D0F2         MOVB *R2+,R3
7C32 06C3         SWPB R3
7C34 06A0         BL   @E1
7C36 7AD0
7C38 C100         MOV  R0,R4
7C3A 0A54         SLA  R4,5
7C3C 06A0         BL   @E1
7C3E 7AD0
7C40 C140         MOV  R0,R5
7C42 A105         A    R5,R4
7C44 06A0         BL   @E1
7C46 7AD0
```

```
7C48 C180        MOV  R0,R6
7C4A 06C6        SWPB R6
7C4C D160        MOVB @>9802,R5
7C4E 9802
7C50 06C5        SWPB R5
7C52 D160        MOVB @>9802,R5
7C54 9802
7C56 06C5        SWPB R5
7C58 0605        DEC  R5
7C5A C0C3   E5   MOV  R3,R3
7C5C 1340        JEQ  F
7C5E 0284        CI   R4,>02FF
7C60 02FF
7C62 1B3D        JH   F
7C64 C004        MOV  R4,R0
7C66 0220        AI   R0,>1800
7C68 1800
7C6A 06C0        SWPB R0
7C6C D800        MOVB R0,@>8C02
7C6E 8C02
7C70 06C0        SWPB R0
7C72 D800        MOVB R0,@>8C02
7C74 8C02
7C76 04C0        CLR  R0
7C78 D020        MOVB @>8800,R0
7C7A 8800
7C7C 132C        JEQ  E2
7C7E 0950        SRL  R0,5
7C80 C044        MOV  R4,R1
7C82 0981        SRL  R1,8
7C84 0AB1        SLA  R1,11
7C86 A040        A    R0,R1
7C88 04C0        CLR  R0
7C8A D012        MOVB *R2,R0
7C8C 0950        SRL  R0,5
7C8E 0220        AI   R0,>03B4
7C90 03B4
7C92 D800        MOVB R0,@>9C02
7C94 9C02
7C96 06C0        SWPB R0
7C98 D800        MOVB R0,@>9C02
7C9A 9C02
7C9C 06C1        SWPB R1
7C9E D801        MOVB R1,@>8C02
7CA0 8C02
7CA2 06C1        SWPB R1
7CA4 0221        AI   R1,>4000
7CA6 4000
7CA8 D801        MOVB R1,@>8C02
7CAA 8C02
7CAC 0200        LI   R0,>0008
7CAE 0008
7CB0 D820   E3   MOVB @>9800,@>8C00
7CB2 9800
7CB4 8C00
7CB6 0600        DEC  R0
7CB8 16FB        JNE  E3
7CBA 0221        AI   R1,>2000
7CBC 2000
7CBE 06C1        SWPB R1
7CC0 D801        MOVB R1,@>8C02
7CC2 8C02
7CC4 06C1        SWPB R1
7CC6 D801        MOVB R1,@>8C02
7CC8 8C02
7CCA 0200        LI   R0,>0008
7CCC 0008
7CCE D806   E4   MOVB R6,@>8C00
7CD0 8C00
7CD2 0600        DEC  R0
7CD4 16FC        JNE  E4
7CD6 0603   E2   DEC  R3
7CD8 0584        INC  R4
7CDA 0582        INC  R2
7CDC 10BE        JMP  E5
7CDE D805   F    MOVB R5,@>9C02
7CE0 9C02
7CE2 06C5        SWPB R5
7CE4 D805        MOVB R5,@>9C02
7CE6 9C02
7CE8 045A        B    *R10
```

Exemple d'appel de la routine "TEXTE"

Note : cette routine doit bien sûr se trouver en mémoire au moment du CALL LINK

```
100 REM PASSAGE EN MODE BITMAP (
    VOIR 99 MAGAZINE NUMERO 7)
110 CALL SCREEN(1)
120 CALL LOAD(-31890,56,0)
130 CALL CLEAR
140 CALL PEEKV(-32766,X,"",-3225
    0,X,"",-31745,X,"",-31741,X,
    "",-31434,X,"",-31229,X)
150 CALL POKEV(6912,208)
160 FOR I=0 TO 2
170 FOR A=0 to 255 STEP 8
180 CALL POKEV(A+256*I+6144,A,A+
    1,A+2,A+3,A+4,A+5,A+6,A+7)
190 NEXT A
200 NEXT I
210 FOR I=0 TO 15
220 CALL LINK("TEXTE","CARACTERE
    S EN MODE BIT MAP",I,2,I*16)
230 NEXT I
240 GOTO 240
```

Transfert de jeux d'aventures de cassette sur disquette

Denise Amrouche

Le module "Adventure" est accompagné de différentes versions de jeux distribués en France, uniquement sur cassettes, d'où la déception des heureux possesseurs de disquettes, obligés de chercher un magnétophone pour trouver leur chemin dans leur aventure préférée.
Le court programme suivant remédie à la situation une fois pour toutes, en vous permettant de transférer le contenu de la cassette sur une disquette.

Mode d'emploi

- mettre, dans DSK1, une disquette initialisée comptant au moins 55 secteurs libres;
- insérer la Mini-mémoire et y charger le programme;
- passer en Basic, OLD CS1 pour charger la cassette désirée. Ne pas s'inquiéter du message d'erreur "I/O ERROR 50";
- taper CALL LINK("TRANSF "), le transfert s'effectue sur la disquette avec comme nom de programme "PI";
- à l'aide du "DISK MANAGER", changer le nom du programme (GHOST, SAVAGE... par exemple).

NOTE : le programme ne fonctionne pas avec "SAVAGE ISLAND Part 2", qui est le jeu le plus gourmand en mémoire.

Détails du programme

Les DATAs du début forment le PAB (bloc d'accès aux périphériques);

- >06 option SAVE Memory Image (voir manuel de l'éditeur/assembleur, page 297);
- >0709 adresse du début du buffer en mémoire vive vidéo;
- >3300 nombre d'octets à transférer;
- >07 longueur du nom ("DSK1.PI");
- >3B00 adresse en mémoire vive vidéo choisie pour y mettre le PAB;
- >6028 = VMBW;
- >6038 = DSRLNK.

Bon jeu.

99

TRANSFERT DE JEUX D'AVENTURE DE CASSETTE SUR DISQUETTE

```
7F00                  AORG >7F00

7F00 0600   P         DATA >0600,>0709,0,>3300,>0007
7F02 0709
7F04 0000
7F06 3300
7F08 0007
7F0A 4453             TEXT 'DSK1.PI '
7F0C 4B31
7F0E 2E50
7F10 4920
7F12 0200   T         LI   R0,>3B00
7F14 3B00
7F16 0201             LI   R1,P
7F18 7F00
7F1A 0202             LI   R2,>11
7F1C 0011
7F1E 0420             BLWP @>6028
7F20 6028
7F22 0206             LI   R6,>3B09
7F24 3B09
7F26 C806             MOV  R6,@>8356
7F28 8356
7F2A 0420             BLWP @>6038
7F2C 6038
7F2E 0008             DATA 8
7F30 045B             B    *R11

701C                  AORG >701C

701C 7F32             DATA >7F32,>7FF8
701E 7FF8

7FF8                  AORG >7FF8

7FF8 5452             TEXT 'TRANSF'
7FFA 414E
7FFC 5346
7FFE 7F12             DATA T

                      END
```

Transfert de cassette à disquette des jeux de Scott Adams

Michel Teyssou

Vous êtes certainement nombreux à apprécier les jeux d'aventures fonctionnant avec le module du même nom. Certes, ils ne sont pas graphiques mais ils sont cependant très intéressants, ne serait-ce que parce qu'ils nous obligent à réviser notre anglais. Hélas, le temps moyen de chargement d'un jeu ne descend jamais en dessous de deux minutes, et peut aller jusqu'à trois minutes ! Je suis sûr que, comme moi, vous hésitez longuement avant d'en charger un, et que quand vous êtes bloqué quelque part dans le déroulement de l'énigme, vous préférez arrêter plutôt que d'en charger un autre.

Maintenant, plus d'excuses, grâce au système que je vais vous indiquer, vous pourrez désormais charger vos jeux en cinq secondes environ.

Vous devez pour cela avoir sous la main :

- la Mini-mémoire ;
- le programme SAVE (disquette E/A Texas) ;
- Le programme DEBUG (disquette E/A Texas).

Démarche

1) mettez la Mini-mémoire ;
2) choisissez l'option '3 MINI MEMORY' ;
3) initialisez-la (option '3 : RE-INITIALISE') ;
4) chargez DEBUG (option '1 : DSK1.DEBUG') ;
5) charger SAVE (option '1 : DSK1.SAVE').

Faites FCTN<QUIT> pour revenir à la mire et choisissez EASY BUG. Avec l'option M ('MODIFY CPU MEMORY'), affichez l'adresse >701F. Vous y trouverez >D8. Il faut y mettre >F0 et valider avec <ENTER>.

Quitter EASY BUG (FCTN <QUIT>) et reprenez l'option '3 MINI MEMORY'. Quand vous y êtes, choisissez l'option '2 RUN', et affichez DEBUG comme nom de programme... <ENTER> (les trois points signifient "appuyez sur <ENTER>").

DEBUG est chargé dans la partie haute de la mémoire de 32Ko (>A000 à >FFFF) alors que SAVE est chargé dans la partie basse (>2000 à >3FFF).

Nous allons nous servir de DEBUG pour modifier certaines parties de SAVE et pour visionner en mémoire vive vidéo les informations chargées à partir de la cassette.

Avec l'option M de DEBUG, affichez le contenu de l'adresse >2BB6. Vous trouverez >1000. C'est l'adresse de départ en mémoire vive vidéo des octets qui vont être transférés, il faut mettre à la place >0709 car c'est à partir de cette adresse que sont chargées toutes les données des jeux (voir article de Denise Amrouche dans le numéro 7 de "99 Magazine").

Appuyez sur <ENTER> pour valider et, toujours avec l'option M, affichez >2CC2. Vous trouverez >0206. Vous le remplacez par >0460. Appuyez sur "espace" pour valider et afficher l'adresse suivante >2CC4 ; vous y trouvez >0000. Il faut y mettre >2CEC et valider avec <ENTER>. On va réaliser la même opération avec >2D2C... >2D2C = >81CA, remplacez par >0460 puis "espace" : >2D2E = >1302, remplacez par >2D64 et <ENTER>.

Nous y sommes presque, il faut maintenant transférer le PAB (bloc en mémoire vive vidéo où sont inscrites les informations relatives au fichier disquette) de >0F80 en >0340, car >0F80 se trouve à l'intérieur de la zone où vont s'inscrire les données de la cassette.

Avec l'option M :

- affichez >2CEE... >2CEE = >0F80 mettre >0340...
- affichez >2D00... >2D00 = >0F89 mettre >0349...
- affichez >2D66... >2D66 = >0F89 mettre >0349...

Voilà, le programme SAVE est opérationnel, il ne lui manque plus qu'une information que nous lui transmettrons après le chargement du fichier cassette.

Faites FCTN<QUIT> et choisissez 'TI-BASIC'. Certains jeux ont tendance à empiéter sur la zone réservée aux disquettes (partie numéro 2 de "SAVAGE ISLAND", par exemple). Pour parer à toute éventualité, faites CALL FILES(1)... NEW... puis OLD CS1... et chargez le contenu de la cassette sans tenir compte du message d'erreur en fin de chargement.

Vous êtes en Basic, faites (sans numéro de ligne) CALL LINK ("DEBUG") et <ENTER>. Cette ligne va remonter de deux positions et rien ne semble se passer. C'est normal, vous êtes en Basic et, pour que les codes ASCII soient utilisables, il faut leur ajouter >60. Appuyez sur la lettre <U> qui est la commande de DEBUG pour réaliser cette fonction. Un point apparait, DEBUG est prêt à fonctionner avec le Basic.

Vous pouvez vous servir de DEBUG pour lister les données inscrites en, mémoire vive vidéo à partir de la cassette.
.M 709V,800...
La deuxième adresse est l'adresse d'arrêt. Vous pouvez choisir n'importe laquelle entre >709 et >3FFF. La lettre V indique que c'est en mémoire vive vidéo. Les données numériques qui se trouvent entre les textes correspondent à des aiguillages d'adresses pilotés par le programme principal logé dans le module.

Nous allons, avec DEBUG, repérer la fin des données cassette et calculer le nombre total d'octets à enregistrer. Ce chiffre en hexadécimal nous est indispensable. Vous affichez toujours avec l'option M :
2E00V,3F00...

Tous les jeux ont des données qui dépassent l'adresse >2E00 et qui, pour la majorité s'arrêtent entre >2E40 et >2FF0. Certaines descendent beaucoup plus bas ; ce sera à vous de le déterminer.

Les chiffres commencent à défiler. Vous pouvez interrompre ce défilement avec la barre d'espacement et la reprendre de même. A un moment donné, vous n'allez plus rencontrer que des zéros. Il faut alors repérer la première adresse libre après la dernière donnée (prendre une adresse paire). Vous pouvez sortir de toutes les commandes de DEBUG en faisant FCTN<X>. Faites-le et, quand le point apparait, utilisez la commande H et affichez :
XXXX,709... (XXXX) représente la première adresse libre que vous venez de trouver. Les deux chiffres vont être additionnés, divisés, multipliés par DEBUG mais le seul résultat qui nous intéresse est celui de la soustraction. Il va vous donner le nombre d'octets en hexadécimal que nous allons enregistrer sur la disquette.

Avec l'option M, affichez le contenu de l'adresse >2BBA :
2BBA=2000. Mettez à la place YYYY qui est le nombre d'octets que vous a donné H1-H2. Appuyez sur <ENTER>. Sortez de DEBUG avec l'option <Q> et <ENTER>. Attention, ne faites jamais FCTN<QUIT> car vous effaceriez toutes les données qui se trouvent en mémoire vive vidéo.

Vous êtes revenu au Basic ; sur la ligne du dessus, il y a votre commande d'accès à DEBUG. Faites la même chose pour SAVE :
CALL LINK("SAVE")... Ne vous inquiétez pas du résultat ; c'est normal. Toujours à cause du Basic, on ne peut pas visionner les codes ASCII sans leur ajouter >60. Comme SAVE ne comporte pas de commande qui permette de réaliser cette opération, on va travailler à l'aveuglette !

Le curseur est le trait blanc qui se trouvent à gauche et en bas de l'écran. Tapez :

DSK1.XXXXXXXXXX.
En cas de doute, vous pouvez revenir en arrière avec FCTN<S>. (XXXXXXXXXX) représente, bien sûr, le nom de dix lettres maximum que vous avez choisi de donner au programme. Appuyez sur <ENTER> et SAVE va transférer les données qui se trouvent en mémoire vive vidéo sur la disquette (n'oubliez pas de mettre une disquette de travail !).

Si vous avez commis une erreur et que SAVE ne fonctionne pas, sortez-en avec FCTN<BACK>. Revenez à DEBUG et vérifiez à nouveau le contenu de vos adresses.

Si vous avez plusieurs jeux à charger, faites FCTN<QUIT> entre chaque opération pour réinitialiser la mémoire vive vidéo afin de déterminer avec précision la fin des nouvelles données que vous aller charger. Cela n'effacera pas SAVE et DEBUG qui se trouvent en CPU. Il ne vous restera plus qu'à déterminer le nombre d'octets du nouveau programme et à l'inscrire en >2BBA à la place de l'autre.

Tout cela peut paraître compliqué et fastidieux, mais le résultat en vaut la peine. N'hésitez pas à réaliser ce transfert, vous ne le regretterez pas.

A titre d'exemple, voici les données de jeux que je possède. Si vous en avez d'autres, écrivez-moi !

SAVAGE ISLAND
PART1 de >709 à >2E44, >273B octets
PART2 de >709 à >37D2, >30C9 octets

PYRAMID OF DOOM
De >709 à >2E8A, >2781 octets

THE GOLDEN VOYAGE
De >709 à >2EF4, >27EB octets

GHOST TOWN
De >709 à >2E42, >2739 octets

PIRATE ADVENTURE
De >709 à >2F80, 2877 octets

Michel Teyssou
Arbus
64230 Lescar

Transfert rapide d'informations sur cassettes

Alexandre Amortila

La routine en langage machine ici proposée permettra aux lecteurs pressés de réaliser un gain de temps de l'ordre de t/3 en transférant les images Bit-Map (ou toutes autres données situées en mémoire vive vidéo) vers une cassette ou inversement, et ce, en appliquant les conseils de Gérard Baroni et Jean Marin, qui ont fait un grand travail sur la haute résolution et le TI-99/4A.

Les sous-programmes s'appellent de la même manière "LI" et "SI" et ils nécessitent une bonne qualité de cassette. Le source est riche en commentaires, ce qui devrait permettre une compréhension aisée. Il comprend en plus les deux sous-programmes "CS1 ON" et "CS1OFF", dont la présence est appréciable.

N.D.L.R.: Dans sa configuration actuelle, la routine transfère 14336 octets (>3800) à partir de l'adresse 0 (en mémoire vive vidéo). Il est toutefois possible de modifier ces paramètres en "POKant", après chargement en mémoire de la routine, le nombre d'octets à l'adresse 32000 (>7D00) et l'adresse de base du bloc à transférer à l'adresse 32002 (>7D02). Par exemple, pour transférer un bloc de mémoire de 1000 octets (>3E8) situé à l'adresse 2200 (>898), il faut employer l'instruction :

CALL LOAD (32000,3,232 ,8,152)

puisque 3 * 256 + 232 = 1000 et 8*256+152=2200.

99

Source des routines de transfert rapide vers une cassette ("SI"), depuis une cassette ("LI"), de mise "on" ("CN") et de mise "off" ("CF") du moteur du magnétocassette.

```
                ****************************************
                * SI, LI, CS1ON, CS1OFF                *
                * Ne pas oublier de mettre les titres *
                * dans la table des references.        *
                ****************************************

7D00                 AORG >7D00

7D00 3800       TA   DATA >3800       Octets a transferer
7D02 0000            DATA 0           Adresse de debut de
                *                     transfert
7D04 02E0       SI   LWPI >83E0
7D06 83E0
7D08 0201            LI   R1,TA
7D0A 7D00
7D0C 04C0            CLR  R0
7D0E 0202            LI   R2,>300
7D10 0300
7D12 0208            LI   R8,>1E19    Code de SBZ 25
7D14 1E19
7D16 0203            LI   R3,21
7D18 0015
7D1A 06A0            BL   @>13BA
7D1C 13BA
7D1E 1D16            SBO  22          Motor 1 on
7D20 0200            LI   R0,>13E2
7D22 13E2
7D24 0300            LIMI 1           Interruption
7D26 0001
                *                     niveau 1 : le TMS
                *                     9900 effectue un
                *                     BLWP @>0004 a
```

```
                  *                              chaque interruption
                  *                              generee par le
                  *                              TMS 9901
7D28 04C4  A6   CLR  R4
7D2A 0690       BL   *R0
7D2C 0602       DEC  R2
7D2E 16FC       JNE  A6
7D30 0704       SETO R4
7D32 0690       BL   *R0
7D34 C105       MOV  R5,R4
7D36 06C4       SWPB R4
7D38 0690       BL   *R0
7D3A C105       MOV  R5,R4
7D3C 06C4       SWPB R4
7D3E 0690       BL   *R0
7D40 0202  A9   LI   R2,8
7D42 0008
7D44 04C4  A7   CLR  R4
7D46 0690       BL   *R0
7D48 0602       DEC  R2
7D4A 16FC       JNE  A7
7D4C 0704       SETO R4
7D4E 0690       BL   *R0
7D50 D7E0       MOVB @>83F5,*R15
7D52 83F5
7D54 0202       LI   R2,>40
7D56 0040
7D58 D7CA       MOVB R10,*R15
7D5A 04C7       CLR  R7
7D5C 04C4  A8   CLR  R4
7D5E D12F       MOVB @>FBFE(R15),R4
7D60 FBFE
7D62 A1C4       A    R4,R7
7D64 0690       BL   *R0
7D66 0602       DEC  R2
7D68 16F9       JNE  A8
7D6A C107       MOV  R7,R4
7D6C 0690       BL   *R0
7D6E 022A       AI   R10,>40
7D70 0040       /
7D72 0605       DEC  R5
7D74 16E5       JNE  A9
7D76 10FF  B0   JMP  B0            Boucle d'attente
           *                       d'une interruption
7D78 0460       B    @B1
7D7A 7E96
           ****
7D7C 02E0  LI   LWPI >83E0
7D7E 83E0
7D80 0201       LI   R1,TA
7D82 7D00
7D84 43A0       SZC  @>1344,14
7D86 1344
7D88 0200       LI   R0,>4000
7D8A 4000
7D8C 0203       LI   R3,25
7D8E 0019
7D90 06A0       BL   @>13BA
7D92 13BA
7D94 1D16       SBO  22            Motor 1 on
7D96 C1CA       MOV  R10,R7
7D98 04C0       CLR  R0
7D9A D820       MOVB @>1443,@>837C
7D9C 1443
7D9E 837C
7DA0 0208  D3   LI   R8,>7530
7DA2 7530
7DA4 0300       LIMI 1
7DA6 0001
7DA8 0206       LI   R6,C1
7DAA 7DB0
7DAC 0203  C5   LI   R3,25
7DAE 0019
7DB0 0241  C1   ANDI R1,>FF
```

```
7DB2 00FF
7DB4 0608      DEC  R8
7DB6 136F      JEQ  B1
7DB8 0202      LI   R2,30
7DBA 001E
7DBC C000      MOV  R0,R0
7DBE 1601      JNE  B8
7DC0 A082      A    R2,R2
7DC2 06A0  B8  BL   @>1572
7DC4 1572
7DC6 1001      JMP  C0
7DC8 10F3      JMP  C1
7DCA 0602  C0  DEC  R2
7DCC 16FA      JNE  B8
7DCE 0209      LI   R9,>7FFF
7DD0 7FFF
7DD2 0208      LI   R8,8
7DD4 0008
7DD6 33C9      LDCR 9,15
7DD8 1E00      SBZ  0
7DDA 1D03      SBO  3
7DDC 06A0  C4  BL   @>15BA
7DDE 15BA
7DE0 1001      JMP  C3
7DE2 10FC      JMP  C4
7DE4 0608  C3  DEC  R8
7DE6 16FA      JNE  C4
7DE8 1D00      SBO  0
7DEA 37C3      STCR 3,15
7DEC 6243      S    R3,R9
7DEE C0C9      MOV  R9,R3
7DF0 0A29      SLA  R9,2
7DF2 A0C9      A    R9,R3
7DF4 0963      SRL  R3,6
7DF6 0263      ORI  R3,1
7DF8 0001
7DFA 020A      LI   R10,C7
7DFC 7E08
7DFE 0283      CI   R3,17
7E00 0011
7E02 11D4      JLT  C5
7E04 0460      B    @>1580
7E06 1580
7E08 06A0  C7  BL   @>1572
7E0A 1572
7E0C 10FD      JMP  C7
7E0E 0202      LI   R2,7
7E10 0007
7E12 06A0  C8  BL   @>1572
7E14 1572
7E16 10CC      JMP  C1
7E18 0602      DEC  R2
7E1A 16FB      JNE  C8
7E1C 0206      LI   R6,D9
7E1E 7E4E
7E20 C000      MOV  R0,R0
7E22 1621      JNE  C9
7E24 D820      MOVB @>1424,@>837C
7E26 1424
7E28 837C
7E2A C007      MOV  R7,R0
7E2C 04C7      CLR  R7
7E2E 06A0      BL   @>15A0
7E30 15A0
7E32 8105      C    R5,R4
7E34 1A30      JL   B1
7E36 C144      MOV  R4,R5
7E38 0585      INC  R5
7E3A 0507      NEG  R7
7E3C 06A0      BL   @>15A0
7E3E 15A0
7E40 162A      JNE  B1
7E42 100D      JMP  D1
7E44 0247  E1  ANDI R7,>FF
```

```
7E46 00FF
7E48 0507        NEG  R7
7E4A 06A0        BL   @>15A0
7E4C 15A0
7E4E C145   D9   MOV  R5,R5
7E50 1322        JEQ  B1
7E52 0220   D4   AI   R0,>40
7E54 0040
7E56 D7E0        MOVB @>83E1,*R15
7E58 83E1
7E5A 1000        NOP
7E5C D7C0        MOVB R0,*R15
7E5E 04C7   D1   CLR  R7
7E60 0605        DEC  R5
7E62 169E        JNE  D3
7E64 1015        JMP  D6
7E66 0202   C9   LI   R2,>40
7E68 0040
7E6A 04C7        CLR  R7
7E6C 06A0   E0   BL   @>15A0
7E6E 15A0
7E70 06C4        SWPB R4
7E72 23A0        COC  @>1344,14
7E74 1344
7E76 1607        JNE  D7
7E78 712F        SB   @>FBFE(R15),R4
7E7A FBFE
7E7C 1306        JEQ  D8
7E7E 0285        CI   R5,1
7E80 0001
7E82 1303        JEQ  D8
7E84 10E4        JMP  D9
7E86 DBC4   D7   MOVB R4,@>FFFE(R15)
7E88 FFFE
7E8A 0602   D8   DEC  R2
7E8C 16EF        JNE  E0
7E8E 10DA        JMP  E1
7E90 D820   D6   MOVB @>1438,@>837C
7E92 1438
7E94 837C
7E96 43A0   B1   SZC  @>1344,14
7E98 1344
7E9A 43A0        SZC  @>32,14
7E9C 0032
7E9E 1E03        SBZ  3
7EA0 1D0C        SBO  12
7EA2 1D01        SBO  1
7EA4 1D02        SBO  2
7EA6 1D16        SBO  22          Motor 1 off
7EA8 0460        B    @>70
7EAA 0070
            ****
7EAC 04CC   CN   CLR  R12         Debut "CS1ON"
7EAE 1D16        SBO  22
7EB0 04C0   CC   CLR  R0
7EB2 D800        MOVB R0,@>837C
7EB4 837C
7EB6 045B        RT
            ****
7EB8 04CC   CF   CLR  R12         Debut "CS1OFF"
7EBA 1E16        SBZ  22
7EBC 10F9        JMP  CC

7FE0             AORG >7FE0
7FE0 5349        TEXT 'SI    '
7FE2 2020
7FE4 2020
7FE6 7D04        DATA >7D04
7FE8 4C49        TEXT 'LI    '
7FEA 2020
7FEC 2020
7FEE 7D7C        DATA >7D7C
```

```
7FF0 434E       TEXT 'CN    '
7FF2 2020
7FF4 2020
7FF6 7EAC       DATA >7EAC
7FF8 4346       TEXT 'CF    '
7FFA 2020
7FFC 2020
7FFE 7EB8       DATA >7EB8

701E            AORG >701E
701E 7FE0       DATA >7FE0

*************************************
* Le TMS 9901 est le generateur d'in- *
* terruptions dans le TI-99. Pour re- *
* gler la frequence des interruptions *
* il faut lever le bit du controle    *
* (0) du CRU a 1, ce qui met le TMS   *
* 9901 en mode horloge et nous permet *
* d'ecrire ou de lire le contenu du    *
* registre d'horloge a l'aide de LDCR *
* ou STCR. Ce contenu est represente  *
* par les bits de 2-15 du CRU. Ce re- *
* gistre est decremente a chaque cy-  *
* cle d'horloge, et quand il atteint  *
* (0) il est reinitialise a sa valeur *
* initiale, et une interruption est   *
* generee. Dans l'etat actuel de la   *
* ROM du TI, et puisque le branche-   *
* ment se fait a l'adresse >4 qu'on   *
* ne peut pas modifier, il est diffi- *
* cile d'accelerer les programmes LI  *
* et SI au dela d'une limite ou le    *
* TMS 9900 n'aura pas le temps d'ef-  *
* fectuer son travail entre deux in-  *
* terruption.                         *
*************************************
   END
```

Courrier des Lecteurs

Ne pensez-vous pas que l'informatique des loisirs et du vrai amateur se situe plus chez les petits ordinateurs huit bits, avec un langage assembleur encore élémentaire et du Basic simple qui permet de créer des programmes "Do it yourself", que du côté de la nouvelle génération Macintosh et de la copie Atari 520ST qui sont tellement puissants et professionnels que leur perfection tue l'esprit du bricoleur ? Ils sont livrés sans Basic et s'adressent à des gens qui consomment des logiciels tout faits et achètent des souris et autres extensions très cher parcequ'il ne savent pas programmer. L'informatique amateur finira peut-être un jour comme la passion des radio-amateurs ; les radios transistors sont devenus si bon marché et si complexes qu'on ne les construit plus soi-même comme par le passé. La photo amateur a connu le même sort ; aujourd'hui, les agrandissements des photographies en couleurs sont plus difficiles à faire que les tirages noir et blanc d'autrefois, et nécessitent un équipement important parceque la photo s'est perfectionnée, alors que les copies couleurs sont très bon marché dans les grandes surfaces ce qui fait que très peu d'amateurs font encore leurs tirages couleurs eux-mêmes. Ainsi, la micro informatique avec des programmes écrits en Basic était intéressante tant que l'amateur apprenant le Basic avait plus de chance qu'un utilisateur de bureau ignorant. L'amateur cherche souvent à faire le maximum avec des frais minimums, en y mettant de l'enthousiasme. Le TI-99 avec son Basic, son générateur de sons à trois voix et bruits, ses seize couleurs et ses lutins, permet de créer plus facilement des programmes soi-même que le Commodore 64 qui oblige à taper d'interminables séries de chiffres à "POKer", ou de consommer des programmes commerciaux.

H.-H. Umstätter - Genève

Il convient tout d'abord de faire deux petites mises au point : le TI-99/4A n'est pas un ordinateur 8 bits puisque le TMS 9900 est un processeur 16 bits et l'assembleur qui l'équipe est loin d'être élémentaire puisqu'il s'agit d'une version modifiée (affichage 2*40 colonnes) de celui qui équipe les mini-ordinateurs de la marque. Ceci n'est bien sûr pas valable si vous parlez de l'assembleur ligne par ligne de la mini-mémoire.

Il est vrai que les micro-ordinateurs sont de plus en plus complexes mais il est encore parfaitement possible, pour un amateur, de programmer un Macintosh (en fait, et contrairement à ce que l'on pourrait penser, il y a beaucoup de points communs entre un TI et un Macintosh).

Petites annonces (gratuites)

Ventes

Vends TI-99/4A, Basic étendu, manuels en français, module "Treasure Island", cordon magnéto, manettes de jeux, 5 cassettes "99 Magazine", nombreux jeux sur cassettes, nombreux livres : 1500 FF (valeur réelle : 2700 FF).

Bruno Alves - 4, allée de Champagne - 78170 La Celle St Cloud - tél.: 39.69.75.11.

Vends ou échange Rotor Raider (150 FF), Ant Eater (150 FF). Achète (ou échange) Popeye, Pole Position, Speech Synthesizer et modules américains Miner 2049, Pacman, Computerwar, Jumpman Junior, Story Machine, Congo Bongo, Baseball, Blasto, Hustle, Black Jack and Poker, Touch Typing Tutor, Munch Mobile, Physical Fitness, Sneggit.

Philippe Tenand - 2 bis, rue de l'Egalité - 94300 Vincennes - Tél.: 808.17.25.

Vends TI-99/4A, boîtier d'extension avec lecteur et contrôleur de disquettes, extension de mémoire 32Ko, Basic étendu, huit modules de jeux, manettes de jeux, prix à débattre. Offre divers livres sur le TI et des programmes sur disquettes.

Alain Leydier - 43 bis, rue Jean Jaurès - 42150 La Ricamarie - Tél.: 77.80.28.75.

Vends TI-99/4A (1000 FF), câble magnéto (80 FF), module Basic étendu avec manuel français (550 FF), modules "Lunar Lander" (80 FF), "Solar System" (100 FF), "Parsec" (200 FF), manettes de jeux (120 FF). Le tout 2000 FF avec livres.

Pierre-Yves Billard - 11, carrefour Prudhomme - 91000 Bondoufle - Tél.: 60.86.10.93.

Vends TI-99/A4 (12/83), Basic étendu, joysticks Texas et câble magnéto : 1700 FF.
Boîtier d'extension, mémoire 32Ko, lecteur de disquettes, disquettes, Editeur/assembleur et boîte de rangement pour disquettes : 5700 FF.
Interface Centronics, imprimante GP 50A et divers (papier, rubans, etc.) : 1800 FF.
Modules (Parsec, Star Trek, Super Demon Attack, ...), 4 cassettes Epsilon Software (Sun Games, ...), 5 cassettes de programmes divers, Rubis sacré, adaptateur pour joysticks Atari et Joystick Gun Shot Vulcan : 1000 FF.
Livres sur le TI: Boîte à outils, à l'affiche, 102 programmes, 99 Magazine, Shift 3: 100 FF.

J-M. Sarrobert - 90, avenue des Montagnys - Les Bougeries - 74200 Allinges - Tél.: (50) 71.71.00 poste 470 (heures de bureau).

Vends boîtier d'extension, carte 32Ko, contrôleur de disquettes, lecteur de disquettes : 4500 FF. RS232C : 1200 FF. Seïkosha GP100A : 1500 FF. Modules de jeux, crayon optique, Editeur/assembleur, U.C., Basic étendu, l'ensemble : 9000 FF.

Gérard Cauvin - impasse du Quartier - Le Bosquet - 30200 Bagnols sur Ceze - Tél.: (66) 89.83.92 (HR).

Vends TI-99/4A bon état (Décembre 83), câble magnéto, prise péritel, Basic étendu + manuel, Mini-mémoire, nombreux programmes en cartouches : Startrek, Mash, Parsec, TI Invaders, Chisholm trail, Tombstone city, Soccer et nombreux programmes sur cassettes et "99 Magazine" 1 à 10 : 3700 FF.

Olivier Guerret - 113, rue du Colonel de Rochebrune - 92500 Rueil-Malmaison - Tél.: 751. 88.59.

Vends modules Basic étendu + manuel en français : 500 FF. FootBall : 120 FF. Blasto : 120 FF.

T. Martel - 12, rue Bonne Santé - 76620 Le Havre - Tél.: 35.45.30.58.

Vends TI-99/4A, boîtier d'extension, carte mémoire 32Ko, carte RS232, contrôleur de disquettes, lecteur de disquettes interne, synthétiseur vocal, manettes de jeux, imprimante GP 100A, modules Mini-mémoire, Foot, Echecs, Music Maker, Speech Editor, Basic Etendu, Editeur/Assembleur, cassettes de jeux, nombreux livres, numéros 1 à 9 de 99 Magazine.

Jean-Claude Hocquet - 9, allée de la tonnelle - 59650 Villeneuve d'Ascq - Tél.: 20.91.63.58 après 17H30.

Vends TI-99/4A état neuf, manettes de jeu, magnétophone avec câble, 11 modules de jeux (Aventure, Echecs, Blasto, Parsec, Star Treck, etc.), Basic par soi-même, "Les techniques de programmes de jeux" 1 et 2, 99 Magazine 1 à 9, le tout 2000 FF (à débattre).

C. Boismoreau - 11, rue de la Clef - 75005 Paris - Tél.: 45.71.59.69 aux heures de bureau.

Vends boîtier d'extension périphérique, contrôleur et lecteur de disquettes, module Disk

Manager, disquettes, carte RS232/Centronics d'origine, câble RS232/Imprimante type Seïkosha, Mini-mémoire avec cassette et livre de Denise Amrouche "Initiation au langage assembleur", gestion de fichiers, livre "Pratique de l'ordinateur familial Texas Instruments" niveau 1 et niveau 2.

Philippe Mechelaere - 06, rue des Saints Sauveurs - 92260 Fontenay aux Roses - Tél.: 43.50.88.18 après 19H ou Sam/Dim toute la journée.

Vends TI-99/4A, Basic étendu, Mini-mémoire, Editeur/assembleur, Logo, Basic Graphic, extension 32Ko, interface Para., synthétiseur, six modules de gestion, trente modules de jeux, livres et revues, moniteur noir et blanc, imprimante Seïkosha GP50. L'ensemble : 8000 FF, ou détaillé.

Jean Reibel - 9, square V. Fleming - 92350 Le Plessis Robinson - Tél.: 631.46.11.

Vends TI-99/4A parfait état (12/83), magnéto, manettes, extension 32Ko, sortie parallèle avec leurs divers cordons et câbles. Modules Basic étendu, Fichiers, Statistiques, Mini-mémoire, Connect Four, Othello, Echecs, Parsec, avec leurs notices. Cassettes Basic par soi-même, Extended Basic, Aide à la programmation N° 1, Line by Line Assembler, Cassettes 99/A, B, C, D et E, 9 cassettes avec la revue "99", livres divers. Le tout - valeur 9170 FF - bradé 4000 FF. Urgent.

François Dodivers - 330, allée des Nids - Domaine du Cap Sicié - 83500 La Seyne sur mer - Tél.: 94.25.18.09.

Vends module Basic étendu fabriqué aux USA avec le manuel complet permettant sa parfaite utilisation. Etat neuf, peu servi, Le tout pour 750 FF environ à débattre.

Patrick Giraud - 3, clos des mésanges - 78280 Guyancourt - Tél.: 30.44.11.75 après 18h.

Vends TI-99/4A avec extension 32Ko, neuf jeux d'arcades (dont un inédit en France), deux jeux de reflexion, Basic étendu, Mini-mémoire, magnétophone, plusieurs cassettes (contenant des jeux, des programmes de calculs...) et de nombreux livres : série des "99 Magazine", Initiation au Basic, langage assembleur, Basic étendu... Alimentation et câbles de liaisons inclus. Prix envisagé : 3800 FF.

Yves-Luc Perrève - 3, allée des Marronniers - 91940 Les Ulis - Tél.: 69.28.37.15.

Vends TI-99/4A + manettes de jeux + Parsec + Munchman + TI Invaders + Burger Time + Echecs + "99 Magazine" + livres : 2000 FF. Mini-mémoire : 500 FF. Synthétiseur de voix + Speech Editor : 800 FF. Tunnels of Doom : 300 FF. Pirate's Island : 200 FF.

M. Lailheugue - 30, domaine de Bel Abord - 91380 Chilly Mazarin - Tél.: 64.48.61.81.

Vends, le tout en excellent état, TI-99/4A Péritel, magnéto Philips Data 6600 avec cordon, boîtier avec câble interface, extension 32Ko, interface RS232, lecteur de disquettes, contrôleur et module DISK MANAGER.
Modules Basic Etendu, LOGO 2, Early learning fun, Beginning grammar, Zero Zap, Aligator mix, Music Master, Number magic, Personal record keeping.
Cassettes le Basic par soi-même, Basic étendu, jeux d'entreprise, 99/C, 99/D, 99/7, revues "99 Magazine" et diverses, manettes de jeux.
Etudie toutes propositions.

M. Beguin - R.N. 96 - Napollon - 13400 Aubagne - Tél.: 42.03.73.60 heures bureau.

Vends TI-99, Mini-mémoire, Basic étendu, dix modules, cassettes Texas Instruments, livre Minimem et une vingtaine de livres pour TI, 200 programmes environ sur cassettes, tous les câbles. 3900 F possible sur quatre mois.

Jean-Pierre Diot - 47, rue Jacquart - 51100 Reims. Tél.: 26.02.69.48.

Achats

Cherche (échange ou achat) modules, programmes (Basic simple ou étendu). Réponse assurée.

Thierry Debran - 7, rue de la Fontaine - 12000 Rodez - Tél: (65) 42.05.32 (vers 20 h).

Achète, séparement ou ensemble, console TI-99/4A, prise Péritel, alimentation, module diagnostique (urgent), modules divers (Statistiques, etc.), cassettes Test de math 1 et 2, Test de physique 1, techniques et programmes de jeux (avec livrets), boîtier d'extension, cartes interfaces série et parallèle, contrôleur de disquettes, lecteur de disquettes, imprimante, disquettes diverses, MODEM.

Michel Voisin - 22, route de Venerque - 31190 Auterive - Tél.: 61.08.21.88.

Recherche imprimante d'occasion pour un enfant ayant un TI-99/4A.

Alain Barrau - 101, rue Saint-Lazare - 75009 Paris - Tél.: 42.85.79.25.

Achète cartouches de jeux. Cherche, en particulier, jeu d'Echecs.

Fabrice Roussot - 2, résidence Vincennes - 77330 Ozoir-la-Ferrière.

Divers

Recherche schéma de mémoire vive extérieure pour TI-99/4A.

Philippe Escaich - Lescure - 09420 Rimont - Tél.: (61) 96.31.95.

Abonné "99 Magazine" souhaite connaître possesseur TI-99/4A région Orléans.

Jean-Noël Mérour - 12, rue de la Corne - 45650 Saint-Jean-Le-Blanc - Tél.: 38.66.73.96.

99

CASSETTE D'ACCOMPAGNEMENT DISPONIBLE

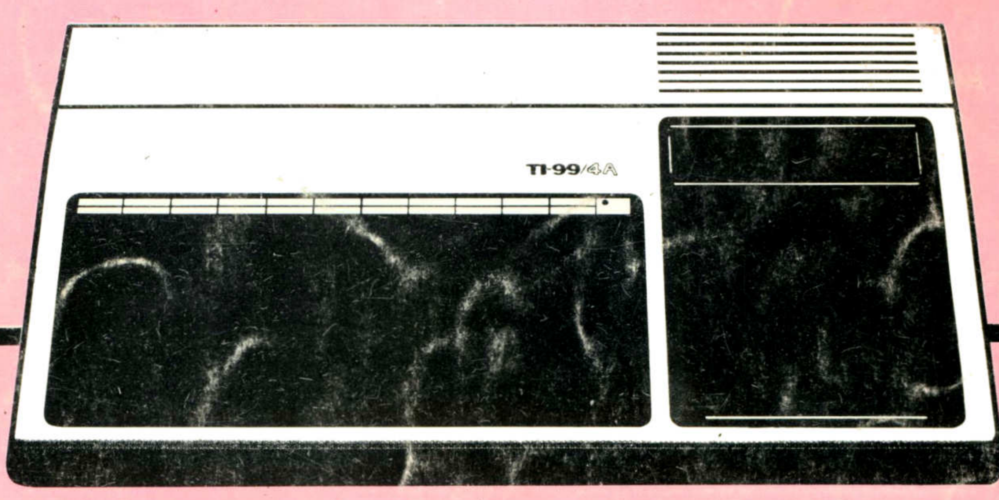
99 MAGAZINE

Fenêtrage

Copie d'écran

Connecteur multiple

Indentation automatique



Numéro 11 — Trimestriel — 40 Francs

ISSN 0759-9005

Sommaires des numéros 1 à 10 de "99 Magazine"

Numéro 1

Musik Maker à l'essai	
Pourquoi acheter un TI-99/4A	
A vous de programmer	
Déballage du TI-99	
Les avantages du Basic étendu	BE
Initiation au Basic	BTI
Jeu du Pendu	BTI
Fabriquez votre joystick	
Appelez moi KEY	BTI/BE
Programmes d'aide à la décision	BE
Jeu de l'Alphabet	BTI
Le club d'utilisateurs TI-Education	
Introduction au LOGO	L
Les tours de Hanoï	BTI
Récupérez les programmes Apple	BTI/BE
Le troisième menuet de Bach	BTI
Le dernier robot	BTI
Cherchez l'ancêtre	BTI
Gestion personnelle de fichiers	
Quand le TI devient un harmonium	BTI
Fichiers en Basic	BE

Numéro 2

Micro-monde LOGO : les lutins	L
La bataille navale	BTI
Le crocodile savant	
Jeu du Simon	BTI/BE
Jeu des carrés	BTI
L'ordinateur de l'avenir	
TI-Writer à l'essai	
Canon de Purcell	BE
Chicken Helper	BE
Sauvegarde des fichiers et prog.	BTI/BE
Le Mastermind	BTI
Othello	BTI
Astéroïdes	BE
Kermit la grenouille	BE
Le Biorythme	BTI

Numéro 3

Un programme de calcul	BE
Annonce de Texas Instruments	
Othello à l'essai	
Apprendre Basic en programmant (1)	BTI
Introduction au Pascal	P
Puissance 4	BE
Les modes d'adressage en Assembleur	A
Géographie : la carte de France	BTI
Créez votre touche RESET	
Puzzle	BTI
Awari	BTI
Le sapin de Noël	BE
PARSEC à l'essai	
LOGO, seulement pour enfant ?	L
Le Mastermind en assembleur	A
Du nouveau sur les cassettes	
Le magnétophone TI	

Numéro 4

SOS Hélicoptère	BE
Utilisations de la Mini-mémoire	BTI
Communiqué de Texas Instruments	
Le serveur de restaurant	BTI
"Echecs" à l'essai	
Combat de chars	BTI
Réalisez votre cordon de magnéto	
Apprendre Basic en programmant (2)	BTI
Jeu de mémorisation	BTI
Jack-pot	BE
Les nombres en LOGO	L
Graphisme en haute résolution	BTI + A
Sous-programmes en assembleur	A
Horloge	BE
Un détecteur de lumière pour le TI	
Le Club M.T.I.	
Le système P-UCSD	P
Les lutins du Basic étendu	BE
Le solitaire	BTI
Claustrophobia	BTI
"DISPLAY" et "ACCEPT" avec PRK	BTI
Une erreur dans le manuel MINIMEM	
Réparez votre assembleur ligne par ligne	A

Numéro 5

Conversions	BTI
Systèmes linéaires	BTI
Structure des disques du P-Système	P
Poèmes et graphismes	BTI
Un désassembleur en assembleur	A
Les tours de Hanoï	BE
Trouvez le bon mot	BTI
Le jeu de la vie	BTI ou A
Programmathèque	
Les opérateurs logiques	BE
Apprendre Basic en programmant (3)	BTI
Bibliographie	
Warangal	BTI
Graphisme en haute résolution	BTI + A
Le trésor de l'Armada	BE
Ecriture littérale d'un nombre	P
Copie d'écran sur imprimante Epson	BTI
Description du port d'entrées/sorties	
Translet	BE
Le Club Ordinabis	
Le Club Ticazur	

Numéro 6

Des chiffres et des lettres	BTI
A l'heure du Pascal	P
Kong	BE
Mini-mémoire moins "mini"	
Edipage	BTI
Caractères ASCII/page graphique	BTI + A
Musik	BTI
Copie d'écran sur Seïkosha GP-100	BE
Nouvelles fonctions avec MMEM	BTI + A
Le Basic à la loupe	BTI ou BE
Star Treck	BE
Ecriture directe dans registres VDP	BTI
My tailor was rich	BTI
Le code de la route	BE
Un double face sur votre TI	

Numéro 7

La défausse	BTI
Le tableur TI-Calc	
Sauvegarde d'images graphiques	A
Copie d'écran	A
Copie d'écran sur Seïkosha GP-250	BE
La compilation séparée	P
Les bibliothèques	P
Trouvez votre chemin dans les jeux d'aventure TI	A
Srand	BTI
Galaxia	A
LOGO et la récursivité	L
Dessinez avec le Basic Etendu	BE
Accès au mode BitMap en Basic TI	BTI
Music Box Dancer	BTI
Des nouvelles du Club Ticazur	
Routines graphiques pour le BE	BE + A

Numéro 8

Bit-Map et Mini-mémoire	BTI + A
Le train sifflera trois fois...	BE
Comptes familiaux	BE
Mission spéciale	BE
Crayon optique	A
Le traitement des fichiers en Pascal	P
Editeur de caractères	BE
Kaleïdoscope	BE
Opérations	BTI

Numéro 9

Transfert image/cassette	BTI + A
Ticrok	TI
Pointeurs et variables en Pascal	P
Gestion de fichier en Basic étendu et en assembleur	BE ou A
Un nouveau module graphique	
Musique et Mini-mémoire	BTI + A
Le module Maximem	
Pentaxe	BTI

Numéro 10

Un puzzle de chiffres	BTI
Quatre nouvelles routines pour le crayon optique	A
Tracé de courbes en haute résolution par splines cubiques	BTI + A
Expérience pédagogique orthographe et informatique	BE
Barmélo(s)	BTI
La disquette "Fichier d'adresses"	
Des caractères dans la page graphique haute résolution	BTI + A
Transfert de jeux d'aventures de cassette sur disquette	A
Transfert de cassette à disquette des jeux de Scott Adam's	A
Transfert rapide d'informations sur cassettes	A

* **BTI** : Basic TI. **BE** : Basic étendu. **A** : Assembleur. **P** : Pascal. **L** : Logo.

Tous les numéros sont disponibles.

99 Magazine n° 11

Éditions MEV - 64/70, rue des Chantiers - 78000 Versailles Directeur de la publication : Hervé Thiriez

Sommaire

Éditorial	Hervé Thiriez	5
La guerre des fourmis	J.-P. Tellier	6
Harpagon	Christian Badaroux	8
Haute résolution en temps réel	Alexandre Amortilla	16
Alligator	Bernard Legrand	23
Flag	Jean-Pierre Lartillier	26
Texas Aliens	Christian Lucas	31
Un connecteur multiple	Jean Penné	33
Copie d'écran pour le Basic Étendu	Michel Teyssou	35
Indentation automatique	Gérard Santraille	38
Le château hanté	François-Xavier Nelva	45
Dessins en Basic TI	Jean-Pierre Dhée	51
Fenêtrage	Jean-Philippe Guillemant	54
Disques double face	Yann Fasquel	61
Petites annonces		62
Le club "SHER-TI"		63

Bon de commande

Cassettes de jeux

99A	Mineur, Prêt, Electronique		à	35,00 F	
99B	Yahtzee, Chardef, Division, Régression linéaire		à	35,00 F	
99C	Dames, Nim, Division 2, Isola, Schmoo, Robots		à	35,00 F	
99D	Conjugaisons, Caractérologie, Car Driver, Poker, Cannibales		à	35,00 F	
99E	Mic-Math, Course de chevaux, Poursuite, Jeu de dés, Guerre atomique, Course en ligne droite		à	35,00 F	

Disquettes de jeux

Mastermind (cf. 99 Magazine n° 3)		à	55,00 F	
Galaxia (cf. 99 Magazine n° 7)		à	55,00 F	

Disquettes utilitaires

Routines graphiques (cf. 99 Magazine n° 4, 5, 6 et 7)		à	55,00 F	
Désassembleur (cf. 99 Magazine n° 5)		à	55,00 F	
Crayon optique (cf. 99 Magazine n° 8)		à	55,00 F	

Anciens numéros

99 Magazine 1 2 3 4 5 6 7 8 9		à	25,00 F	
Cassette d'accompagnement 1 2 3 4 5 6 7 8 9		à	35,00 F	

Numéros 10 et 11

99 Magazine 10 11		à	40,00 F	
Cassette d'accompagement 10 11		à	55,00 F	

Total TTC

Supplément pour port avion hors CEE
15,00 F par revue et/ou cassette et/ou disquette

Montant du règlement joint

Nom : ..

Adresse : ..

Envoyer ce bon et votre règlement à :

Editions MEV - 64, rue des Chantiers - 78000 Versailles

Éditorial

Comme nous l'avions annoncé dans le précédent numéro de "99 Magazine", vous avez entre les mains le dernier numéro de l'unique revue francophone entièrement consacrée au TI-99/4A Texas Instruments.

Nous remercions les Lecteurs qui ont proposé des solutions propres à prolonger la vie de la revue, certains allant jusqu'à nous dire de faire passer le prix - sans cassette - à 100 Francs, ce qui, avouons-le, eut été exagéré ! Si l'on en croit votre courrier, la plupart des Lecteurs de "99 Magazine" ont 'appris' l'informatique en s'aidant des articles de nos collaborateurs et auteurs et, dans la plupart des cas, la revue a servi de tremplin vers d'autres matériels, plus "professionnels" ; nous pouvons donc considérer que nous avons rempli notre mission.

Ce numéro, bien qu'étant l'ultime, n'a pas pour autant été négligé : vous pourrez trouver des jeux avec ***J.-P. Tellier*** *qui vous propose* la guerre des fourmis, ***Christian Lucas*** *et son excellent* Texas Aliens *et* ***François-Xavier Nelva*** *qui nous a concocté un jeu d'aventure baptisé le* château hanté. *Encore des jeux, mais plus éducatifs :* flag *de* ***Jean-Pierre Lartillier*** *et, pour les tout-petits,* Alligator *de* ***Bernard Legrand. Christian Badaroux*** *nous confie un programme de comptabilité familiale qu'il a baptisé* Harpagon *et qui présente la particularité, par rapport à ce qui a été publié jusqu'à présent, de ne pas requérir d'autre extension que le Basic Étendu.* ***Gérard Santraille****, après sa petite infidélité du numéro 10, s'est remis au Pascal pour écrire un remarquable programme d'*indentation automatique. *Plus 'pointus' puisque faisant appel à l'assembleur, un programme de* copie d'écran *pour le Basic Étendu écrit par* ***Michel Teyssou****, des routines permettant de dessiner en* haute résolution *et* en temps réel *avec le Basic Étendu, proposé par* ***Alexandre Amortilla*** *et, enfin, une participation de* ***Jean-Philippe Guillemant*** *qui sacrifie à la mode du* fenêtrage. *Au chapitre de la 'bidouille' très utile, nous trouvons le* connecteur multiple *de* ***Jean Penné*** *qui autorise la connection simultanée de plusieurs modules, ces derniers étant reconnus individuellement par le système d'exploitation de notre TI. Nous ne conclurons pas sans citer* Dessins en Basic TI, *une 'fantaisie graphique' de* ***Jean-Pierre Dhée****, ainsi que les précisions apportées par* ***Yann Fasquel*** *sur l'utilisation des* disques double face *sur le TI-99/4A.*

À bientôt, peut-être, sur un autre matériel...

Hervé Thiriez

Rédacteur en chef - Directeur de la publication : Hervé Thiriez.

Comité de rédaction : Jean-Luc Bazanegue, Gérard Santraille.

Ont collaboré à ce numéro : Alexandre Amortilla, Christian Badaroux, Jean-Luc Bazanegue, Jean-Pierre Dhée, Yann Fasquel, Jean-Philippe Guillemant, Jean-Pierre Lartillier, Christian Lucas, Bernard Legrand, François-Xavier Nelva, Jean Penné, Gérard Santraille, J.-P. Tellier, Michel Teyssou.

Éditions MEV - 64/70, rue des Chantiers - 78000 Versailles - Tél. : 39.51.24.43.

Impression : Atelier graphique du centre - 25, rue Michel Lecomte - 75003 Paris - Tél. : 42.72.96.19.

La guerre des fourmis

J.-P. Tellier

Nous vous proposons un court programme de jeu, écrit en Basic Étendu, dont le scénario est très simple : une fourmi bleue (en l'occurence vous) est prisonnière des fourmis rouges. Pour trouver la porte de la Liberté, notre fourmi doit manger au moins dix 'portions magiques'

Les points

Le but du jeu étant bien sûr d'en marquer le plus possible, voici la répartition des points :

- un point pour une fourmi rouge ;
- 100 points pour une 'portion magique' ;
- 100 points pour l'évasion ;
- 40 points pour un 'Boni' (un Boni correspond à la 'consommation' de 20 portions magiques, plus l'évasion).

Les déplacements de la fourmi bleue sont obtenus grâce à la manette de jeux numéro 1.

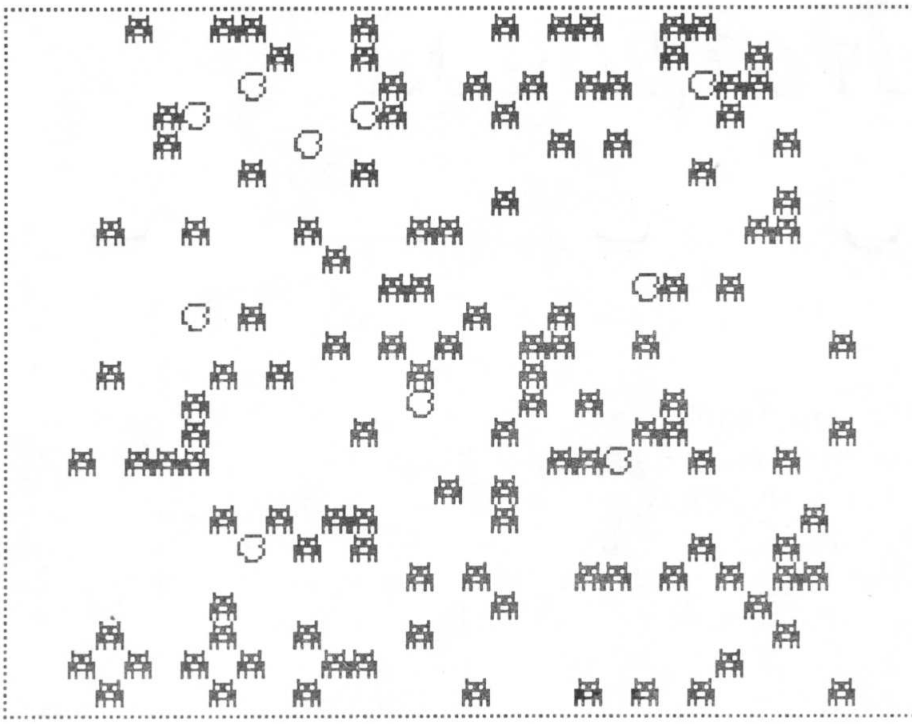
99

Programme "La guerre des fourmis"

Basic Étendu

```
190 RANDOMIZE
200 CALL CLEAR :: CALL SC
   REEN(2):: FOR I=1 TO 8
    :: CALL COLOR(I,16,1)
   :: NEXT I
210 GOTO 1370
220 CALL SCREEN(2)
230 CALL CHAR(96,"427E5AF
   FE7FFA5A5")
240 CALL CHAR(111,"00")
250 CALL CHAR(112,"3E4181
   8281C1223C")
260 CALL COLOR(11,16,2)
270 CALL CHAR(104,"427E5A
   FFE7FFA5A5")
280 CALL CHAR(136,"FF8181
   81818181FF")
290 CALL COLOR(14,3,2)
300 CALL COLOR(9,6,2)
310 CALL COLOR(10,10,2)
320 X=16
330 FOR J=1 TO 24
340 CALL HCHAR(J,3,111,28
   )
350 NEXT J
360 Y=12
370 GOT=0
380 CALL HCHAR(Y,X,96)
390 COUNT=1
400 CALL JOYST(1,DX,DY)
410 A=X
420 B=Y
430 IF (DX=0)+(DY=0)=-2 T
   HEN 540
440 X=X+DX/4
450 Y=Y-DY/4
460 IF (X<3)+(X>30)+(Y<1)
   +(Y>24)<0 THEN 1120
470 CALL HCHAR(B,A,111)
480 CALL GCHAR(Y,X,Z)
490 CALL HCHAR(Y,X,96)
500 IF Z=111 THEN 540
510 IF Z=104 THEN 890
520 IF Z=112 THEN 1020
530 IF Z=152 THEN 1200
540 R=INT(24*RND)+1
550 C=INT(28*RND)+3
560 CALL GCHAR(R,C,Z)
570 IF Z=111 THEN 600
580 IF Z=96 THEN 660
590 GOTO 540
600 CALL HCHAR(R,C,104)
610 IF COUNT<>20 THEN 630
620 GOSUB 930
630 CALL SOUND(100,110,1)
640 COUNT=COUNT+1
650 GOTO 400
660 CALL CLEAR :: CALL SC
   REEN(2)
670 CALL SOUND(2500,440,2
   ,659,5,880,10,-6,15)
680 DISPLAY AT(5,1):"LA F
   OURMI BLEUE EST MANGEE
   !"
690 DISPLAY AT(8,1):"LES
   ROUGES ETAIENT ";COUNT
    :: DISPLAY AT(10,1):"
   LES PORTIONS MANGEES "
   ;GOT
720 SCORE=GOT*100+COUNT
730 DISPLAY AT(13,1):"VOT
   RE POINTAGE EST ";SCOR
   E
740 DISPLAY AT(15,1):"LE
   VIEUX RECORD EST ";HIS
   CORE
750 IF HISCORE>SCORE THEN
    790
760 HISCORE=SCORE
770 DISPLAY AT(18,1):"FEL
   ICITATION !" :: DISPLA
   Y AT(20,1):"POUR LE NO
   UVEAU RECORD."
790 DISPLAY AT(23,1):"VOU
   LEZ-VOUS REJOUER (O/N)
   ?"
810 CALL KEY(0,KEY,S)
820 IF S=0 THEN 810
830 CALL CLEAR :: CALL SC
   REEN(2)
840 IF (KEY=79)+(KEY=111)
   THEN 320
850 !
860 CALL CLEAR :: CALL SC
   REEN(2)
870 DISPLAY AT(12,4):"MER
   CI D'AVOIR JOUER." ::
   GOSUB 2000 :: FOR D=1
   TO 300 :: NEXT D
880 STOP
890 CALL CLEAR :: CALL SC
   REEN(2)
900 CALL SOUND(2500,440,2
   ,659,5,880,10,-8,2)
910 DISPLAY AT(5,1):"C'ES
   T UN SUICIDE !"
920 GOTO 690
930 FOR I=1 TO 10
940 RO=INT(24*RND)+1
950 CO=INT(28*RND)+3
960 CALL GCHAR(RO,CO,Z)
970 IF Z=96 THEN 940
980 IF Z=112 THEN 940
990 CALL HCHAR(RO,CO,112)
1000 NEXT I
1010 RETURN
```

```
1020 GOT=GOT+1
1030 CALL SOUND(1000,262,
   2,330,2,392,2)
1040 IF GOT<>10 THEN 540
1050 GOSUB 930
1060 RO=INT(24*RND)+1
1070 CO=INT(28*RND)+3
1080 CALL GCHAR(RO,CO,Z)
1090 IF Z<>111 THEN 1060
1100 CALL HCHAR(RO,CO,152
   )
1110 GOTO 540
1120 FOR JOT=1 TO 6
1130 CALL SCREEN(7)
1140 CALL SCREEN(2)
1150 CALL SOUND(10,-5,1)
1160 NEXT JOT
1170 X=A
1180 Y=B
1190 GOTO 650
1200 RESTORE 1240
1205 FOR IK=1 TO 19
1210 READ NHL
1220 CALL SOUND(100,NHL,1
   )
1230 NEXT IK
1240 DATA 262,330,392,523
   ,392,523,330,392,523,6
   59
1250 DATA 523,659,392,523
   ,659,784,659,784,784
1270 CALL CLEAR :: CALL S
   CREEN(2)
1280 BONUS=0
1290 DISPLAY AT(5,1):"LA
   FOURMI EST HORS DE DAN
   GER"
1300 IF GOT<>20 THEN 1320
1310 BONUS=4000
1320 BONUS=BONUS+1000
1330 SCORE=GOT*100+COUNT+
   BONUS
1340 DISPLAY AT(13,1):"VO
   TRE POINTAGE EST ";SCO
   RE
1350 GOTO 740
1360 END
1370 DISPLAY AT(9,6):"LA
   GUERRE" :: DISPLAY AT(
   12,13):"DES" :: DISPLA
   Y AT(15,16):"FOURMIS"
   :: GOSUB 2000
1372 FOR D=1 TO 300 :: NE
   XT D
1374 CALL CLEAR
1380 DISPLAY AT(12,1)BEEP
   :"VOULEZ-VOUS LE GUIDE
    (O/N)?"
1390 CALL KEY(0,K,S)
1400 IF S=0 THEN 1390
1410 IF K=78 THEN CALL CL
   EAR :: GOTO 220
1420 IF K<>79 THEN 1390
1430 CALL CLEAR
1440 DISPLAY AT(3,3):"* S
   OIS LE BIENVENU ! *" :
```



```
   : DISPLAY AT(6,3):"TU
   ES LA FOURMI BLEUE." :
   : DISPLAY AT(8,3):"MAI
   S TU ES PRISONNIERE,"
1450 DISPLAY AT(10,3):"DE
   S FOURMIS ROUGES." ::
   DISPLAY AT(12,3):"POUR
   TROUVER PORTE JAUNE"
```



```
1460 DISPLAY AT(14,3):"TU
   DOIS MANGER PLUS DE"
   :: DISPLAY AT(16,3):"1
   0*PORTIONS MAGIQUES*."
1470 DISPLAY AT(19,3):"TU
   DOIS T'EVADER VITE,"
   :: DISPLAY AT(21,3):"C
   AR LA MORT T'ATTEND !"
1480 DISPLAY AT(24,3):"*T
   APEZ CLE POUR POINTS*"
1670 CALL KEY(0,K,S)
1680 IF S=0 THEN 1670
1690 CALL CLEAR
1700 DISPLAY AT(2,9):"LES
   POINTS" :: DISPLAY AT
   (4,3):"1      POUR UNE
   ROUGE"
1710 DISPLAY AT(6,3):"100
    POUR UNE PORTION" :
   : DISPLAY AT(8,3):"100
   0  POUR L'EVASION"
1720 DISPLAY AT(10,3):"40
   00  POUR BONI"
1730 DISPLAY AT(13,3):"LE
   BONI" :: DISPLAY AT(1
   5,8):"C'EST 20 PORTION
   S" :: DISPLAY AT(17,3)
   :"MAGIQUES + L'EVASION
   ."
1740 DISPLAY AT(20,3):"UT
   ILISEZ LA MANETTE #1"
   :: DISPLAY AT(23,3):"*
   TAPEZ CLE POUR JOUER*"
1930 CALL KEY(0,K,S)
1940 IF S=0 THEN 1930
1945 CALL CLEAR
1950 GOTO 220
2000 RESTORE 2020
2010 FOR I=1 TO 50 :: REA
   D F :: CALL SOUND(170,
   F,1):: NEXT I :: RETUR
   N
2020 DATA 110,139,165,185
   ,196,185,165,139,110,1
   39,165
2030 DATA 185,196,185,165
   ,139,147,185,220,247,2
   62,247
2040 DATA 220,185,110,139
   ,165,185,196,185,165,1
   39,165
2050 DATA 208,247,277,147
   ,185,220,247,220,196,1
   85,175
2060 DATA 165,147,139,123
   ,110,110
```

Harpagon

Christian Badaroux

Ce programme qui, comme son nom l'indique, est un programme de comptabilité (familiale) qui ne requiert, comme extension, que le Basic Étendu et un (ou deux) magnétophone. Afin qu'il fonctionne sans problème avec cette configuration (très proche de la configuration de base), le programme ne comporte pas d'indication quant à la marche à suivre ; c'est pourquoi nous vous proposons une présentation d'"Harpagon" à la manière d'un mode d'emploi.

Finalité

Il y a deux volets principaux :

- la gestion des dépenses et ressources à l'usage d'un foyer avec enregistrement des diverses opérations effectuées et la tenue du solde de votre compte personnel, permettant aussi le rapprochement avec la tenue de votre compte en banque. Il permet de conserver les opérations effectuées et de pouvoir les relire à volonté ;
- la gestion d'un budget prévisionnel et son évolution dans le temps en fonction des dépenses ou revenus engagés.

Termes utilisés

Opérations : signifiera dépenses ou revenus (à usage domestique, les dépenses sont éclatées en 31 chapitres différents, les revenus en une seule tête de chapitre baptisée "REVENUS".

Personnalisation

La dénomination des 31 chapitres est laissée à l'initiative de l'utilisateur, et est effectuée au cours de la phase dite d'INITIALISATION.
Chaque chapitre est prévu pour être affecté à un mode de paiement spécifique, tel que les utilisent les banques, et ventilé en 10 possibilités :

- chèques
- prélèvements
- retrait carte bleue
- virement
- plan d'épargne logement
- amortissement de prêt
- frais divers de banque
- retrait d'espèces
- prélèvement de l'impôt sur le revenu des personnes physiques
- divers

L'affectation d'un mode de paiement spécifique à chaque chapitre est laissée à l'initiative de l'utilisateur.

Abréviations

Prel.nt :	*Prélèvement*
Rt C.B :	*Retrait Carte Bleue*
Vir.ent :	*Virement*
P.E.L :	*Plan d'Épargne Logement*
Am.Pret :	*Amortissement prêt*
Ret.esp :	*Retrait espèces*
I.R.P.P :	*Impôt sur le revenu des personnes physiques*
Solde Cpt.Pers :	*Solde du compte personnel*
Solde Cpt.Ban :	*Solde du Compte Banquaire*
Op. :	*Opérations*

Première utilisation du programme

Le programme prévoit de traiter les données précédemment enregistrées sur une cassette afin de les modifier en fonction de la saisie des données.

Lors de la première utilisation, cette cassette n'a pas été élaborée et il faudra procéder à l'initialisation.

Après avoir fait "RUN", et dès l'apparition du générique, taper "75", ce qui vous conduira au Menu Principal.

L'option numéro 8 vous permettra de procéder à l'initialisation de la description des chapitres, puis des types de paiement, et revenir à l'option "FIN" numéro 9, afin d'effectuer une sauvegarde de ce travail sur une cassette séparée avant de procéder à la suite.

```
    MENU

  1 ETAT INITIAL
  2 BUDGET
  3 TABLEAU FINAL
  4 OPERATION
  5 OP.ET ARCHIVAGE
  6 ETAT DES COMPTES
  7 CONSULT. ARCHIVES
  8 INITIALIS.COR.
  9 FIN

    VOTRE CHOIX ?          9
```

Initialisation

Sur l'option 8 du menu principal apparaît le tableau suivant :

```
    INITIALISATION

  1 ANNEE
  2 NO CHEQUE
  3 SOLDE CPT.PERS.
  4 SOLDE CPT.BANC.
  5 CHAPITRE
  6 TYPE PAYEMENT
  7 CORRECTION
  8 NB ARCHIVES


    VOTRE CHOIX ?          9
```

Année

L'option 1 vous demande deux chiffres correspondant aux deux derniers chiffres de l'année en cours : 86, par exemple, pour 1986.

Lors du changement d'année, il suffit de revenir à ce point pour effectuer la modification. Toutes les opérations entre temps seront affectées à 86 sans avoir besoin de les taper à chaque fois.

Chèques

Chaque chapitre d'opération affecté d'un paiement "CHEQUE" numérote séquentiellement les chèques et ceci sans avoir besoin de taper les numéros correspondants. Lors de l'utilisation d'un nouveau chéquier, revenir à cette option en inscrivant le numéro du futur chèque -1 (numéro du dernier chèque utilisé). Vous pouvez entrer un maximum de sept chiffres.

Solde des comptes

Correspond respectivement au solde de votre compte personnel et compte bancaire.

chapitre

Vous pouvez désigner chaque poste de votre budget par la dénomination de votre choix.
Il faut enregistrer les 31 chapitres consécutivement.

Le numéro en cours apparaît en haut et à gauche de l'écran. Vous pouvez utiliser jusqu'à 12 caractères alphanumériques en majuscules ou minuscules. Vous pouvez vérifier le résultat de cette étape sur l'option 1, 2 ou 3 du Menu Principal.

Type de paiement

A chacun des chapitres précédemment entré, il faut affecter le mode de paiement retenu.

Vérifier surtout que le mode retenu pour les chèques est bien exclusif, si vous voulez que la numérotation automatique des chèques fonctionne sans problèmes. Les options possibles sont récapitulées à la demande de l'option 6 de la phase d'initialisation.

```
  1
  ENTRER CODE TYPE PAYEMENT:

 1 CHEQUES
 2 PREL.NT
 3 RT C.B
 4 VIR.ENT
 5 P.E.L
 6 AM.PRET
 7 FRAIS
 8 RET.ESP
 9 I.R.P.P
10 DIVERS
                                 9
```

Le numéro correspondant au chapitre en question apparaît en haut et à gauche de l'écran. Un exemple d'affectation est joint à cet article, sous la forme d'un tableau.

Option 7 et 8

Pour ces deux dernières options, il convient de se reporter aux paragraphe *Correction* et *Changement de cassette archives*.

Tableau initial

Ce tableau représente l'état de votre budget du mois précédent compte tenu des opérations effectuées précédemment.

Il est le reflet des écarts par rapport aux prévisions. Il va être, pour le mois suivant, modifié chapitre par chapitre suivant la formule :

Tableau final = Tableau Initial + Budget - Dépense (ou revenus)

Tableau final (I) = Tableau Initial (I) + Budget (I) - Dépense (I)

dans lequel I est le numéro du chapitre concerné.

Le tableau initial est égal au tableau final du mois précédent soit :

Tableau Initial (M) = Tableau Final (M-1)

dans lequel M est le mois actuel et M-1 le mois précédent.

Budget

Pour chaque chapitre, c'est la prévision d'engagement de dépenses ou revenus mensuels moyens.

En effet, certaines opérations n'ont pas le même cycle ; des opérations sont engagées mensuellement, d'autres tous les 2, 3, 4, 6 mois, ou annuellement.

La constitution sur la base moyenne mensuelle permet d'effectuer la provision nécessaire correspondante jusqu'au moment ou il faut engager la dépense. Le budget est modifiable à tout moment et il est conservé en données pour éviter d'avoir besoin de l'écrire à chaque saisie.

Le budget peut être équilibré ou non au choix de l'utilisateur. En équilibre, la somme des 31 premiers chapitres doit être égale au 32ème (chapitre "Revenus", qui doit être affecté d'un *signe négatif*.

Tableau final

Tableau final et *tableau initial* mesurent les écarts par rapport aux prévisions en fonction des opérations effectuées (voir le paragraphe *Tableau initial*).

Opérations

L'option 4 du Menu Principal vous permet :

- d'effectuer la saisie d'une opération ;
- d'avoir, par chapitre, la synthèse des opérations effectuées pendant le mois.

Appuyer sur "O" <ENTER> pour faire défiler le numéro du chapitre concerné, afin de pouvoir effectuer la saisie d'une opération sur le chapitre concerné.

Dès la demande d'un chapitre apparaît l'écran suivant :

```
OPERATION
************************


LIBELLE          99

SOMME

DATE

TYPE PAYEMENT:
```

- à droite de OPERATION est rappelé le chapitre concerné. Pour LIBELLE, vous pouvez entrer sept caractères alphanumériques pour résumer le libellé de l'opération ;
- pour la SOMME, celle-ci est limitée aux valeurs extrêmes -1E5 et 1E6 ;
- pour la DATE, entrez le jour (1 à 31), puis le mois (1 à 12) et s'inscrit automatiquement l'année en cours si vous l'avez initialisée au préalable ;
- le type de paiement s'inscrit automatiquement avec éventuellement le numéro du chèque correspondant si l'initialisation a été effectuée au préalable.

En cas d'erreur, il suffit d'entrer un chiffre différent de zéro pour recommencer la saisie de la donnée correspondante.

Après la saisie, le compte personnel est débité ou crédité du montant correspondant, que l'on peut vérifier après la saisie :

```
SOLDE CPT.PERS.-1250

SOLDE CPT.BANC. 0

DIF.  CP-CB   = -1250



    APPUYER SUR UNE TOUCHE
```

Le retour se fait sur le tableau DEPENSES et vous pouvez à nouveau procéder à une nouvelle saisie, comme indiqué en tête de ce paragraphe.

Vous pouvez, sur un même chapitre, saisir plusieurs opérations. La limite du nombre d'opérations est de 30 y compris les opérations non archives du mois précédent.

Une fois la saisie effectuée, vous pouvez sur le tableau DEPENSES, avoir les résultats par chapitre ou faire appel à l'option 5 du Menu Principal pour vérifier l'ensemble des opérations qui sont en mémoire.

Note : l'option 5 permet d'appeler les opérations non soldées du mois précédent.

Synthèse des opérations et archivage

C'est l'option 5 du Menu Principal. Plusieurs présentations vous sont offertes, afin de faire le point et vérifier vos divers comptes.

Les options 1, 2, 3 vous permettent de retirer une opération soldée par votre banque.

Les autres options 4,-5 et 6 sont là pour vérification.

```
    VERIF.OPERATIONS

  1 CLAS.NT PAR DATE
  2 CLAS.NT PAR VAL
  3 CLAS.NT PAR TYPE PAY.NT
  4 TOUTES OP.(ORDRE ENTREE)
  5 VERIF DEP.VALIDEES
  6 VERIF.DEP NON VALIDEES
  7 FIN RETOUR MENU



    VOTRE CHOIX ?          7
```

La durée du classement sur les options 1 et 2 varie en fonction du nombre de données.

Les dates et valeurs sont classées de façon croissante. L'option 3 permet d'appeler les opérations effectuées par le mode de paiement correspondant. Pour l'appel des chèques, c'est le numéro du chèque qui apparaît en place du libellé de l'opération. Sur les 3 premières options, <ENTER> ZERO permet de mettre en archive l'opération correspondante, une confirmation vous sera demandée puis l'état des comptes vous sera donné, et le montant de l'opération sera déduit du compte bancaire, vous permettant ainsi de suivre l'évolution des opérations soldées par votre banque.

L'option 4 ne fait pas la différence entre opérations archives ou pas tandis que 5 et 6 vous permettent de visualiser les opérations effectuées et de savoir où sont classées chacune d'elles.

État des comptes

Les comptes en question sont :

- votre compte personnel ;
- votre compte bancaire.

Il existe toujours un décalage entre la tenue de votre compte et la tenue de votre compte bancaire, tant que la banque n'a pas soldé votre opération.

- votre *compte personnel* est mis à jour chaque fois que vous saisissez une donnée ;
- votre *compte bancaire* est modifié et mis à jour chaque fois que vous archivez une opération sur l'option 1, 2 ou 3 mentionnée au paragraphe précédent.

L'option CORRECTION (qui sera expliquée dans le paragraphe *Correction d'une donnée*) rétablie les soldes des deux comptes en annulant l'opération objet de la correction et remodifie uniquement le compte personnel. Après correction, l'opération ainsi modifiée n'est pas archivée ; il faut suivre la procédure d'archivage concernant cette opération si nécessaire.

Consultation des archives

Il s'agit d'une procédure occasionnelle qui vous permet de lire la cassette "ARCHIVES" comme mentionné plus loin et de visualiser opération par opération.

Chaque archive est numérotée et, pour arrêter le défilement, il suffit d'appuyer sur une touche, ce qui fait apparaître le 'Tableau final' qui est le regroupement par chapitre de toutes les opérations lues précédemment. Cette manipulation permet de faire une synthèse pour éventuellement élaborer un budget plus actuel.

Utilisez éventuellement cette procédure pour mettre la bande de magnétophone au futur point de départ du prochain enregistrement, si vous avez rembobiné la cassette "ARCHIVES" par accident.

Fin de saisie des données

Appelez le numéro 9 du Menu Principal qui va vous donner les instructions de manipulation de cassette afin d'enregistrer les informations.

Première phase

Sur CS1 l'ensemble des données commande :

- les 31 chapitres avec leur dénominateur ;
- les 32 modes de paiement ;
- le nombre d'archives déjà enregistrées ;
- le nombre d'opérations en cours ;
- le numéro du dernier chèque utilisé ;
- les deux chiffres de l'année en cours ;
- les soldes des deux comptes ;
- le tableau final
- le tableau budget

Ensuite suivra l'enregistrement des opérations non soldées qui s'inscrivent sur l'écran.

Un *compte à rebours* vous permet de savoir, en cours d'enregistrement, combien de fichiers restent à être enregistrés.

À la suite de cette opération, il faut changer de cassette de magnétophone ou de cordon pour permettre l'enregistrement des "ARCHIVES" sur CS2.

Seconde phase

La deuxième phase permet de sauvegarder en ARCHIVES les opérations soldées. Deux précautions sont à prendre :

- ne pas rembobiner la cassette "ARCHIVES" afin de ne pas écraser les données précédentes ;
- utiliser le cordon CS2 pour que l'enregistrement se fasse sans problème.

À chaque enregistrement sont visualisées chacune des opérations avec chacune de leurs caractéristiques.

Pour un traitement particulier par un autre programme de votre choix (ou de votre cru), l'enregistrement est composé de la façon suivante :

```
OPEN #1:"CS2",INTERNAL,OUTPUT,FIXED 64
PRINT #1:A,B,C,D,E$
```

dans lequel :

- **A** est la somme de l'opération concernée
- **B** est la date en un seul nombre, sous la forme ANMOJO tel que 12 janvier 86 serait écrit 860112
- **C** est le numéro du chapitre correspondant
- **D** est le numéro de chèque, éventuellement accompagné du code de type de paiement. Exemples :
 - chèque n° 7 281 263 sera écrit 7281263011
 - paiement sous code 3 (retrait C.B) 031
- **E$** libellé tel que saisi, et comportant 7 caractères alphanumériques

Correction d'une donnée

Il s'agit d'une procédure exceptionnelle car, en cours de saisie, vous avez la possibilité de recommencer l'entrée des données. Néanmoins, si vous vous apercevez d'une erreur, il vous faut suivre la procédure suivante :

Menu Principal - option numéro 5, puis option numéro 3 où vous pouvez faire défiler toutes les opérations. Compter le numéro d'ordre de saisie de la donnée à modifier. Si la donnée apparaît en deuxième page au numéro 4, par exemple, le numéro de la donnée sera 13 (9+4), et ainsi de suite. Revenir au menu INITIALISATION puis option 7 CORRECTION.

À la demande de numéro, faire 13 <ENTER> (exemple ci-dessus) et vous devez tombé sur la dépense à modifier. Si ce n'est pas le cas, faire "0" <ENTER>, puis le nouveau numéro de saisie jusqu'à visualiser l'opération concernée.

Taper un chiffre, modifier le libellé, la somme, la date et sur la même ligne, séparé par deux virgules, entrer le numéro du chapitre, le numéro de chèque éventuellement, puis le code du type de paiement (voir le paragraphe *Initialisation*).

- vous tapez 3,7812762,1 pour affecter l'opération corrigée au chapitre 3 paiement par chèque numéro 7812762 et mode de paiement chèque 1.
- vous taperez 5,0,3 pour affectation au chapitre 5 avec un mode de règlement 3 (retrait Carte Bleue).

Les tableaux DEPENSES, les SOLDES DES COMPTES sont corrigés en conséquence et votre opération ainsi corrigée si elle était archivée, se retrouve non archivée. Recommencez la procédure normale si nécessaire.

Précautions d'emploi

- utilisez un magnétophone avec des têtes propres, afin que la lecture des fichiers ne pose pas de problème ;
- évitez de confondre les cassettes :
 - PROGRAMMES
 - DONNÉES COURANTES
 - ARCHIVES

 afin de ne pas écraser les données correspondantes ;
- veillez à ne pas écraser les données précédentes de la cassette "ARCHIVES" en rembobinant la cassette.

Changement de cassette "Archives"

Lorsque la cassette archive est pleine, et afin de repartir sur l'autre face :

- noter sur la cassette le nombre d'archives (consultation archives) ;
- pour repartir à zéro sur l'autre face initialisez, avec l'option 8 du menu INITIALISATION, le nombre à zéro ;
- manipulations spéciales : si vous voulez lire une cassette "ARCHIVES" différente de celle qui est en cours, initialisez NB au nombre correspondant d'archives à lire et, en fin de lecture, rétablir NB à sa valeur initiale.

Note : au fur et à mesure des archivages, il est prudent de noter le nombre d'archives déjà enregistrées pour pouvoir se livrer aux manipulations spéciales indiquées ci-dessus.

99

Programme "Harpagon"

Basic Étendu

```
100 DIM Z(33),U(33),S(33)
   ,G(33),D(30),C(30),B$(
   30),Y(30),X(30),P$(33)
   ,LP(33),DM(
33),M$(20):: DEF VL(I)=Y(
   I)-10*INT(Y(I)/10)
110 DATA MENU,Etat Initia
   l,Budget,Tableau Final
   ,Operation,Op.et Archi
   vage,Etat des Comptes,
   consult. Archives,Init
   ialis.Cor.,Fin,
120 DATA INITIALISATION,A
   nnee,No cheque,Solde c
   pt.Pers.,Solde Cpt.Ban
   c.,Chapitre,Type Payem
   ent,Correction,Nb Arch
   ives,
130 DATA VERIF.OPERATIONS
   ,Clas.nt par date,Clas
   .nt par val,Clas.nt pa
   r type pay.nt,Toutes O
   p.(Ordre entree),Verif
    dep.validees
140 DATA Verif.dep non va
   lidees,fin retour menu
```

```
150 DATA TYPE PAYEMENT:
160 DATA Cheques,Prel.nt,
   Rt C.B,Vir.ent,P.E.L,A
   m.pret,Frais,Ret.esp,I
   .R.P.P,Divers,
170 DATA Somme,Date,ATTEN
   TION,Placer cassette A
   RCHIVES,sur CS1,sur CS
   2,SATURATION,VOUS ALLE
   Z ARCHIVER,Confirmer E
   nter 0 Sinon No
180 DATA Enter No Sinon S
   uite=0,Libelle
190 FOR I=1 TO 53 :: READ
    A$ :: IF (I=2)+(I=3)+
   (I=4)+(I=5)+(I=13)+(I=
   14)+(I=15)+(I=16)+(I=3
   1)+(I>42)THEN J=J+1 ::
    M$(J)=A$
200 NEXT I
210 RESTORE 160 :: FOR I=
   1 TO 10 :: READ LP$(I)
   :: NEXT I220 CALL CHAR
   (128,"FFFFFFFFFFFFFFF
   FFFFFFFFFFFFFFFFFFFF
   FFFFFFFFFFFFFFFFFFFF
   FFFFFF")
230 CALL SPRITE(#1,128,10
   ,74,74,0,5):: CALL MAG
   NIFY(4):: DISPLAY AT(8
   ,8)ERASE ALL:"BADAROUX
   .." :: GOTO 250
240 CALL CLEAR
250 DISPLAY AT(14,8):"HAR
   PAGON" :: DISPLAY AT(1
   6,8):"ENTRER No"
260 ACCEPT AT(16,20):CL :
   : IF CL<>75 THEN 990 E
   LSE CALL DELSPRITE(ALL
   ):: GOTO 420
270 INPUT "NB":NB2 :: GOT
   O 420
280 INPUT "No":I :: IF (I
   <1)+(I>NB)THEN 280 ELS
   E GO SUB 1210 :: GOSUB
    1250 :: GOSUB 530 ::
   CALL C1(0,9,24,27,L)::
    IF L=0 THEN 280
290 CALL SCREEN(5):: DISP
   LAY AT(19,9):"CORRECTI
   ON" :: GOSUB 550
300 DISPLAY AT(22,1):"Cha
   p,No Cheque(ou 0),Typ"
    :: INPUT C(0),Y(0),L
   :: IF (C(0)<1)+(C(0)>3
   2)THEN 300 ELSE IF (L<
   1)+(L>10)THEN 300
310 Y(0)=1E3*Y(0)+10*L ::
    B$(I)=LIBEL$ :: DM(C(
   I))=DM(C(I))-D(I):: X(
   I)=AN*1E4+MO*100+JO ::
    DM(C(0))=DM(C(0))+T :
   : IF VL(I)THEN CB=CB+D
   (I)
320 CALL CLEAR :: CP=CP+D
   (I)-T :: C(I)=C(0):: D
   (I)=T :: Y(I)=10*INT(Y
   (0)/10):: GOSUB 1210 :
   : GOSUB 1250 :: CALL W
   :: GOTO 420
330 INPUT M$(5)&" act. (2
   Ch.) ":AN :: IF (AN<1
   )+(AN>99)THEN 330 ELSE
   420
340 INPUT "Dernier "&M$(6
   )&" utilise (7 Chiffre
   s Max)":CH1 :: IF (CH1
   <99)+(CH1>=1E8)THEN 34
   0 ELSE 420
350 DISPLAY AT(10,2):M$(7
   ):: ACCEPT AT(14,2)VAL
   IDATE(NUMERIC):CP :: C
   ALL AR(CP):: GOTO 420
360 DISPLAY AT(10,2):M$(8
   ):: ACCEPT AT(14,2)VAL
   IDATE(NUMERIC):CB :: C
   ALL AR(CB):: GOTO 420
370 FOR I=1 TO 31 :: DISP
   LAY AT(10,2):I;" Natu
   re Depense(12 Lettres
   Max)" :: ACCEPT AT(15,
   7)SIZE(12):P$(I):: NEX
   T I :: GOTO 420
380 FOR I=1 TO 10 :: DISP
   LAY AT(2*I+3,1):I;LP$(
   I):: NEXT I :: FOR I=1
    TO 31
390 DISPLAY AT(1,2):I;"
   Entrer Code "&M$(9)::
   ACCEPT AT(24,27):LP(I)
   :: IF (LP(I)>10)+(LP(I
   )<1)THEN 390
400 NEXT I :: GOTO 420
410 CALL M(I,2):: ON I GO
   TO 330,340,350,360,370
   ,380,280,270
420 P$(32)="REVENUS" :: L
   P(32)=4 :: P$(33)="TOT
   AUX" :: CALL M(I,1)
430 ON I GOTO 440,450,460
   ,470,630,480,1130,410,
   1030
440 FOR J=1 TO 33 :: Z(J)
   =U(J) :: NEXT J :: GOSU
   B 1310 :: FOR J=1 TO 3
   3 :: U(J)=Z(J):: NEXT
   J :: GOTO 420
450 FOR J=1 TO 33 :: Z(J)
   =G(J) :: NEXT J :: GOSU
   B 1310 :: FOR J=1 TO 3
   3 :: G(J)=Z(J):: NEXT
   J :: GOTO 420
460 FOR J=1 TO 33 :: Z(J)
   =U(J)+G(J)-DM(J):: NEX
   T J :: GOSUB 1310 :: F
   OR J=1 TO 33 :: S(J)=Z
   (J):: NEXT J :: GOTO 4
   20
470 FOR J=1 TO 33 :: Z(J)
   =DM(J):: NEXT J :: GOS
   UB 1310 :: FOR J=1 TO
   33 :: DM(J)=Z(J):: NEX
```

1	Loyer
2	Eau
3	Gaz
4	Electricité
5	Charges
6	Assurance Auto
7	Parking
8	Impôt sur Revenu
9	Taxe habitation
10	Taxe foncière
11	Entretien Auto
12	Essence
13	Habillement
14	Électro-ménager
15	Nourriture
16	Am. Prêt
17	Plan d'épargne
18	Vacances
19	Vins-Alcools
20	Cinéma
21	Cadeaux
22	Liquide
23	C.B. liquide
24	C.B. divers
25	Livres
26	Redevances
27	Cotisations
28	Santé
29	Caisse Épargne
30	Ameublements
31	Divers
32	REVENUS

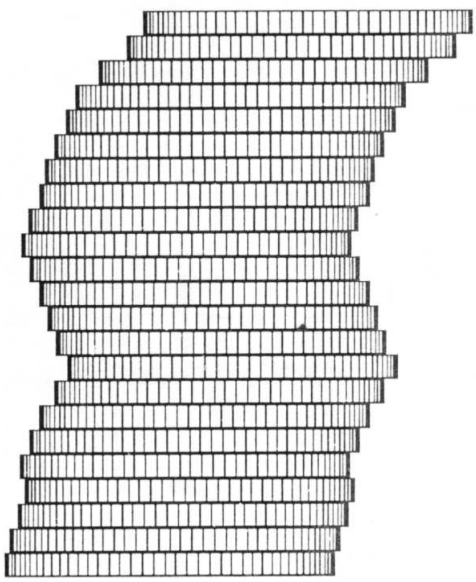
Exemple d'affectations

```
   T J :: GOTO 420
480 CALL C(CB,CP,M$()):: 
   GOTO 420
490 FOR J=1 TO 16 :: DISP
   LAY AT(3+J,1):J :: DIS
   PLAY AT(3+J,5):P$(J)::
    NEXT J :: RETURN
500 FOR J=17 TO 32 :: DIS
   PLAY AT(3+J-16,1):J ::
    DISPLAY AT(3+J-16,5):
   P$(J):: NEXT J
510 IF K=32 THEN CALL SCR
   EEN(16)
520 DISPLAY AT(20,2):"

   " :: DISPLAY AT(21,2):
```

```
P$(33):: RETURN
530 DISPLAY AT(24,3):M$(1
9):: RETURN
540 GOSUB 1220 :: GOSUB 5
50 :: GOTO 580
550 ACCEPT AT(6,19)SIZE(7
):LIBEL$
560 CALL C1(-1E5,1E6,9,19
,T):: CALL AR(T):: IF
K=32 THEN T=-T
570 CALL C1(1,31,12,19,JO
):: CALL C1(1,12,12,22
,MO):: DISPLAY AT(12,2
5):AN :: RETURN
580 IF LP(K)=1 THEN DISPL
AY AT(17,18):CH1+1
590 DISPLAY AT(15,19):LP$
(LP(K)):: GOSUB 530 ::
CALL C1(0,9,24,27,KEY
):: IF KEY<>0 THEN GOT
O 540
600 NB=NB+1 :: B$(NB)=LIB
EL$ :: D(NB)=T :: Z(K)
=Z(K)+T :: C(NB)=K
610 X(NB)=JO+MO*1E2+AN*1E
4 :: IF LP(K)=1 THEN C
H1=CH1+1 :: Y(NB)=1E3*
CH1+10 ELSE Y(NB)=10*L
P(K)
620 CP=CP-T :: CALL C(CB,
CP,M$()):: RETURN
630 T=0 :: CALL M(K,4)
640 LL=0 :: ON K GOTO 660
,650,680,670,650,650,4
20
650 CALL CLAS(Z(),D(),NB)
:: GOSUB 710 :: GOTO 6
30
660 CALL CLAS(Z(),X(),NB)
:: GOSUB 710 :: GOTO 6
30
670 FOR I=1 TO NB :: Z(I)
=I :: NEXT I :: GOSUB
710 :: GOTO 630
680 CALL M(LL,3):: FOR I=
1 TO NB
690 IF INT(Y(I)/10)-100*I
NT(Y(I)/1E3)=LL THEN Z
(I)=I ELSE Z(I)=0
700 NEXT I :: GOSUB 710 :
: GOTO 630
710 T=0 :: I=1 :: L=9
720 CALL CLEAR :: DISPLAY
AT(1,1):"No "&M$(11)&
"    Ch.       "&M$(10)
:: DISPLAY AT(2,1):"*
******** ******* *****
****"
730 IF I>NB THEN L=T :: G
OTO 870 ELSE 740
740 IF Z(I)=0 THEN 930
750 IF K=4 THEN 790
760 IF K<>5 THEN 780
770 IF VL(Z(I))=1 THEN 79
0 ELSE I=I+1 :: GOTO 7
30
780 IF VL(Z(I))=0 THEN 79
0 ELSE I=I+1 :: GOTO 7
30
790 T=T+1 :: DISPLAY AT(2
*T+3,1):USING "#":T ::
IND(T)=Z(I)
800 DISPLAY AT(2*T+3,3):U
SING "##":INT(X(Z(I)))
-100*INT(X(Z(I))/1E2):
: DISPLAY AT(2*T+3,5):
"/"
810 DISPLAY AT(2*T+3,6):U
SING "##":INT(X(Z(I))/
1E2)-100*INT(X(Z(I))/1
E4):: DISPLAY AT(2*T+3
,8):"/"
820 DISPLAY AT(2*T+3,9):U
SING "##":INT(X(Z(I))/
1E4)
830 IF LL=1 THEN DISPLAY
AT(2*T+3,12):USING "##
#####":INT(Y(Z(I))/1E3
):: GOTO 850
840 DISPLAY AT(2*T+3,12):
USING "#######":B$(Z(I
))
850 DISPLAY AT(2*T+3,20):
USING "######.##":D(Z(
I))
860 IF T=9 THEN 870 ELSE
930
870 ON K GOTO 900,900,900
,880,880,880
880 CALL W :: T=0
890 IF I>NB THEN 940 ELSE
I=I+1 :: GOTO 720 ELS
E 880
900 DISPLAY AT(24,1):M$(1
9):: T=0 :: CALL C1(0,
9,24,27,KY)
910 IF KY=0 THEN CALL CLE
AR :: IF I>NB THEN 940
ELSE I=I+1 :: GOTO 72
0
920 IF (KY>L)+(KY<0)THEN
870 ELSE GOSUB 950 ::
GOTO 710
930 I=I+1 :: GOTO 730
940 RETURN
950 GOSUB 970 :: IF KEY<>
48 THEN RETURN
960 CB=CB-D(IND(KY)):: CA
LL C(CB,CP,M$()):: Y(I
ND(KY))=Y(IND(KY))+1 :
: RETURN
970 CALL AF(M$(),12,17,18
)
980 ACCEPT AT(24,27)VALID
ATE(DIGIT):KEY :: KEY=
KEY+48 :: RETURN
990 OPEN #1:"CS1",INTERNA
L,INPUT ,FIXED 64
1000 INPUT #1:AN,CP,CB,NB
,NB2,CH1
1010 FOR I=1 TO 32 :: INP
UT #1:LP(I),P$(I),U(I)
,G(I):: CALL CT(NB+32-
I):: NEXT I
1020 FOR I=1 TO NB :: CAL
L CT(NB-I+1):: INPUT #
1:D(I),X(I),C(I),Y(I),
B$(I):: NEXT I :: CLOS
E #1 :: CALL DELSPRITE
(ALL):: GOTO 420
1030 CALL AF(M$(),12,12,1
4):: FOR I=1 TO 33 ::
S(I)=U(I)+G(I)-DM(I)::
NEXT I
1040 OPEN #1:"CS1",INTERN
AL,OUTPUT,FIXED 64
1050 FOR I=1 TO NB :: IF
VL(I)=0 THEN NB1=NB1+1
1060 NEXT I :: PRINT #1:A
N,CP,CB,NB1,NB2+NB-NB1
,CH1
1070 FOR I=1 TO 32 :: PRI
NT #1:LP(I),P$(I),S(I)
,G(I):: CALL CT(32-I):
: NEXT I
1080 FOR I=1 TO NB :: IF
VL(I)=0 THEN PRINT #1:
D(I),X(I),C(I),Y(I),B$
(I):: GOSUB 1210 :: GO
SUB 1250
1090 NEXT I :: CLOSE #1 :
: CALL AF(M$(),12,13,1
5):: DISPLAY AT(8,12):
"Ne pas Rembobiner" ::
CALL W
1100 OPEN #1:"CS2",INTERN
AL,OUTPUT,FIXED 64
1110 FOR I=1 TO NB :: IF
VL(I)<>0 THEN PRINT #1
:D(I),X(I),C(I),Y(I),B
$(I):: GOSUB 1210 :: G
OSUB 1250
1120 NEXT I :: CLOSE #1 :
: END
1130 CALL AF(M$(),12,13,1
4)
1140 OPEN #1:"CS1",INTERN
AL,INPUT ,FIXED 64
1150 FOR R=1 TO 33 :: Z(R
)=0 :: NEXT R
1160 FOR R=1 TO NB2 :: IN
PUT #1:D(0),X(0),C(0),
Y(0),B$(0):: GOSUB 120
0 :: DISPLAY AT(3,2):N
B2-R+1 :: GOSUB 1250 :
: Z(C(0))=Z(C(0))+D(0)
1170 DISPLAY AT(24,5):"St
op:Presser une touche"
:: CALL KEY(0,MN,MX):
: IF MX=0 THEN 1190
1180 CALL KEY(0,MN,MX)::
IF MX<>0 THEN 1180 ELS
E I=3 :: GOSUB 1310
1190 NEXT R :: CLOSE #1 :
: GOTO 420
```

```
1200 K=C(0):: I=0 :: GOTO
    1230
1210 K=C(I):: GOTO 1230
1220 IF NB>29 THEN DISPLA
    Y AT(10,3)ERASE ALL:M$
    (16):: CALL W :: GOTO
    420
1230 CALL SCREEN(11-5*(K=
    32)):: DISPLAY AT(1,5)
    ERASE ALL:M$(4):: DISP
    LAY AT(1,16):P$(K):: D
    ISPLAY AT(2,5):"******
    ****************"
1240 DISPLAY AT(6,5):M$(2
    0):: DISPLAY AT(9,5):M
    $(10):: DISPLAY AT(12,
    5):M$(11):: DISPLAY AT
    (15,5):M$(9):: RETURN
1250 DISPLAY AT(6,19):B$(
    I):: DISPLAY AT(9,18):
    D(I)
1260 DISPLAY AT(12,18):X(
    I)-100*INT(X(I)/100)::
     DISPLAY AT(12,21):INT
    (X(I)/100)-100*INT(X(I
    )/1E4)
1270 T=INT(Y(I)/10)-100*I
    NT(Y(I)/1E3)
1280 DISPLAY AT(12,24):IN
    T(X(I)/1E4):: DISPLAY
    AT(15,19):LP$(T):: IF
    INT(Y(I)/1E3)>0 THEN D
    ISPLAY AT(17,18):INT(Y
    (I)/1E3)
1290 RETURN
1300 DISPLAY AT(1,5):M$(I
    ):: RETURN
1310 GOSUB 490 :: GOSUB 1
    300
1320 FOR J=1 TO 16 :: DIS
    PLAY AT(3+J,17):USING
    "######.##":Z(J):: NE
    XT J
1330 ON I GOTO 1340,1350,
    1340,1360
1340 CALL W :: GOTO 1370
1350 GOSUB 530 :: CALL C1
    (0,16,24,27,K):: IF (K
    <1)THEN 1370 ELSE CALL
     C1(-1E6,1E6,K+3,18,Z(
    K)):: CALL AR(Z(K))::
    GOTO 1320
1360 GOSUB 530 :: CALL C1
    (0,32,24,27,K):: IF (K
    <1)+(K>16)THEN 1370 EL
    SE GOSUB 540 :: GOTO 1
    310
1370 GOSUB 500 :: GOSUB 1
    300
1380 Z(33)=0 :: FOR J=1 T
    O 32 :: Z(33)=Z(33)+Z(
    J):: NEXT J :: FOR J=1
    7 TO 32 :: DISPLAY AT(
    J-13,17):USING "######
    #.##":Z(J):: NEXT J
1390 DISPLAY AT(21,15):US
    ING "########.##":Z(3
    3)
1400 ON I GOTO 1410,1420,
    1410,1430
1410 CALL W :: RETURN
1420 GOSUB 530 :: CALL C1
    (0,32,24,27,K):: IF (K
    <17)THEN RETURN ELSE A
    CCEPT AT(K-13,18):Z(K)
    :: GOTO 1380
1430 GOSUB 530 :: CALL C1
    (0,32,24,27,K):: IF K<
    17 THEN RETURN ELSE GO
    SUB 540 :: GOTO 1370
1440 SUB CLAS(Z(),X(),NB)
1450 DISPLAY AT(12,4):"Pa
    tientez je classe"
1460 FOR I=1 TO NB :: Z(I
    )=0 :: NEXT I :: R=0 :
    : MN=-1E8
1470 FOR J=1 TO NB :: MX=
    1E8
1480 FOR I=1 TO NB :: IF
    X(I)=MN THEN 1490 ELSE
     1520
1490 FOR K=1 TO NB :: IF
    Z(K)=I THEN 1510
```



```
1500 NEXT K :: R=R+1 :: Z
    (R)=I
1510 IF R=NB THEN 1570 EL
    SE 1550
1520 IF X(I)<MN THEN 1550
1530 IF X(I)>MX THEN 1550
1540 MX=X(I):: L=I
1550 NEXT I
1560 MN=MX :: R=R+1 :: Z(
    R)=L :: NEXT J
1570 CALL CLEAR :: SUBEND
1580 SUB C1(MN,MX,I,J,K)
1590 ACCEPT AT(I,J)BEEP V
    ALIDATE(NUMERIC):K ::
    IF (K<MN)+(K>MX)THEN 1
    590
1600 SUBEND
1610 SUB W
1620 DISPLAY AT(24,3):"Ap
    puyer sur une Touche"
    :: CALL KEY(0,KEY,ST):
    : IF ST=0 THEN 1620
1630 SUBEND
1640 SUB C(CB,CP,M$())::
    CALL SCREEN(10):: DISP
    LAY AT(6,1)ERASE ALL:M
    $(7);CP :: DISPLAY AT(
    10,1):M$(8);CB
1650 DISPLAY AT(14,1):"DI
    F.  CP-CB  = ";CP-CB :
    : CALL W :: CALL SCREE
    N(4):: CALL CLEAR :: S
    UBEND
1660 SUB AR(MX):: MX=1E-2
    *INT(100*MX):: SUBEND
1670 SUB M(I,J):: CALL CL
    EAR :: ON J GOTO 1680,
    1690,1700,1710
1680 RESTORE 110 :: GOTO
    1720
1690 RESTORE 120 :: GOTO
    1720
1700 RESTORE 150 :: GOTO
    1720
1710 RESTORE 130 :: GOTO
    1720
1720 CALL SCREEN(4-10*(J=
    2)):: READ A$ :: DISPL
    AY AT(1,4):A$
1730 FOR J=1 TO 10 :: REA
    D A$ :: IF A$="" THEN
    1750 ELSE DISPLAY AT(2
    *J+2,2):J;A$
1740 NEXT J
1750 DISPLAY AT(24,4):"Vo
    tre Choix ?" :: ACCEPT
     AT(24,27)VALIDATE(DIG
    IT):I :: IF (I<1)+(I>=
    J)THEN 1750
1760 CALL CLEAR :: SUBEND
1770 SUB AF(M$(),MO,JO,MX
    )
1780 CALL SCREEN(7):: DIS
    PLAY AT(8,2)ERASE ALL:
    M$(MO):: DISPLAY AT(12
    ,2):M$(JO):: DISPLAY A
    T(16,2):M$(MX):: SUBEN
    D
1790 SUB CT(I):: DISPLAY
    AT(16,14):I :: SUBEND
```

Haute résolution en temps réel

Alexandre Amortilla

La haute résolution graphique du TI-99/4A reste encore un de ses aspects les plus intéressants. Le programme que nous vous proposons ici (garanti sans 'bugs') requiert le Basic Étendu, une extension 32Ko et un lecteur de disquettes.

Chargement

Les routines sont chargées par :

```
CALL LOAD("DSK1.SCRO")
```

à condition, bien sûr, d'avoir donné "SCRO" comme nom de code objet au moment de l'assemblage.

La bonne idée serait de mettre le programme (LOAD) suivant sur la disquette 1 :

```
10 CALL INIT:: DISPLAY
  AT(18,8)ERASE ALL:"Un
  instant SVP...":: CALL
  LOAD("DSK1.SCRO")
```

Après chargement, on se trouve automatiquement en CALL FILES(1) avec la définition des touches suivantes :

- FCTN-6 : passage en mode haute résolution graphique ("SCR2") ;
- FCTN-5 : retour au mode 32*24 ("SCR1").

À cet instant, on a à notre disposition les fonctions suivantes :

- CALL LINK("SCR2") passage en mode BitMap ;
- CALL LINK("SCR1") retour au mode texte. Dans ce mode, les instructions graphiques sont sans effet.
- CALL LINK("POINT", couleur,y,x) ;
- CALL LINK("CERCLE", couleur,y centre,x centre, rayon) ;
- CALL LINK("LIGNE", couleur,y,x[,y',x']) ; si y' et x' ne sont pas fournis, la ligne tracée démarre de la fin de la ligne précédente jusqu'à y,y, ou bien de l'origine fixée par 'ORIGIN' jusqu'à y,x ;
- CALL LINK("ORIGIN",y,x) pour fixer l'origine de la ligne suivante ;
- CALL LINK("CLEAR") pour effacer l'écran graphique.
- CALL LINK("SAUVE", "DSKn.nomdessin") ;
- CALL LINK("CHARGE", DSNn.nomdessin").

Contraintes

Comme chacun de nous l'imagine, la haute résolution graphique sur le TI et avec le Basic Étendu n'est obtenue qu'au prix de quelques restrictions et contraintes ; il y en a trois :

- tout programme graphique doit démarrer par la ligne suivante :

```
10 CALL LOAD(-31890,56,24)::
CALL LOAD(-31964,56,24)
```

- on est en CALL FILES(1), soit un seul fichier ouvert à la fois ;
- interdiction d'utiliser la directive d'assemblage 'DEF'.

Dessiner sur l'écran du TI en Basic Étendu, avec seulement ces trois restrictions, permet d'admirer l'évolution en temps réel des "Splines" de Gérard Santraille, ce qui, à notre avis, n'a pas de prix.

Le 'source' "SCRS" est intéressant à plusieurs titres :

- on y trouve la routine de dessin de cercles réguliers, d'une façon rapide et optimisée ;
- la routine de tracé de lignes fines, sans double épaisseur pour les lignes obliques ;
- la routine 'DSRLNK' qui manquait cruellement au Basic Étendu (qui contient l'équivalent, mais en GPL).

Interruption du programme

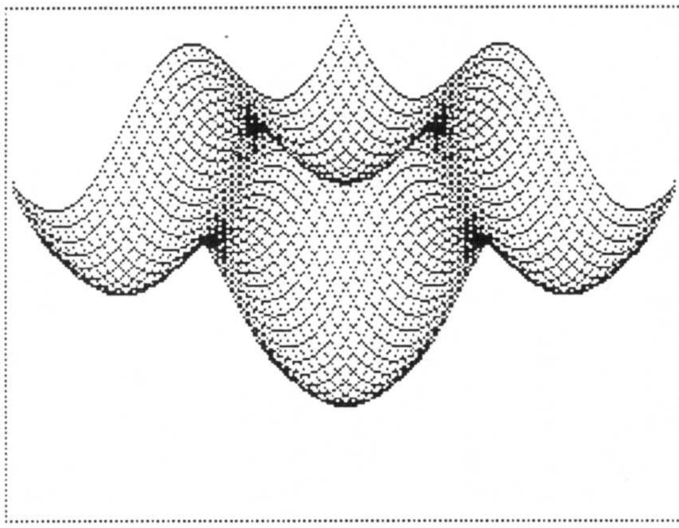
Pour interrompre un programme en cours, appuyer simultanément sur FCTN puis 5 puis 4 : on obtient un BREAK avec retour instantané au mode texte.
Faites la mise au point de vos programmes graphiques en mettant REM avant l'instruction CALL LINK("SCR2"). Une interruption en mode BitMap ne déclenche pas le retour au mode texte. FCTN-4 remet l'écran dans ce mode mais le message d'erreur, très utile pour la mise au point, sera perdu.

99

Source de la routine en assembleur

```
        DEF  SCR1,SCR2,POINT,LIGNE,CLEAR
        DEF  ORIGIN,SAUVE,CHARGE,CERCLE
VSBR    EQU  >2028
VSBW    EQU  >2020
```

```
VMBR    EQU  >202C
VMBW    EQU  >2024
NUMREF  EQU  >200C
XMLLNK  EQU  >2018
STRREF  EQU  >2014
ST      EQU  >837C
*****
BUFF1   EQU  >A000
BUFF2   EQU  BUFF1+>A00
WS1     BSS  32
WS2     BSS  32
COULP   BYTE 0          Color point
COULE   BYTE 0          Color ecran
BITMAP  DATA 0          Flag scr2/scr1
FLAG    DATA 0          1 si appel scr2 par
*****                   clavier:FCTN/6
REG2    DATA >280,>A081,>682,>FF83,>384
        DATA >3685,>786,>1187,0
REG1    DATA >80,>A081,>82,>2083,>84
        DATA >685,>86,>787,0
RVDP1   DATA >E081
*********************************
CL1     BLWP @VSBW
        JMP  $+6
        MOVB 1,@>8C00
        DEC  2
        JNE  $-6
        RT
*****
CLEAR   LWPI WS1
        LI   0,BUFF2
        LI   2,BUFF2+>A00
CL4     CLR  *0+
        C    0,2
        JNE  CL4
        MOV  @BITMAP,@BITMAP
        JNE  CL2
        LI   0,>A00
        LI   2,>E00
        JMP  CL3
CL2     CLR  0
        LI   2,>1800
CL3     CLR  1
        BL   @CL1
        LI   0,>2000
        LI   2,>1800
        BL   @CL1
        B    @RETOUR
*** BIT-MAP ***********
SCR20   MOV  11,@BRET
        MOV  @BITMAP,@BITMAP
        JNE  RETOUR
        SETO @BITMAP
        MOVB @REG2+2,@>83D4
        LI   1,REG2
SCR23   MOV  *1+,2
        JEQ  SCR21
        MOVB 2,@>8C02
        SWPB 2
```



```
        MOVB 2,@>8C02
        JMP  SCR23
SCR21   CLR  0        ----*
        LI   1,BUFF1     *
        LI   2,>A00      *
        BLWP @VMBR       *
        LI   1,BUFF2     *
        BLWP @VMBW ____  *
        LI   0,>1B00
        LI   1,>D000
        BLWP @VSBW
        CLR  1
        LI   2,768
        LI   0,>58
        MOVB 0,@>8C02
        SWPB 0
        MOVB 0,@>8C02
SCR22   SWPB 1
        MOVB 1,@>8C00
        SWPB 1
        INC  1
        C    1,2
        JNE  SCR22     _________
        MOVB @RVDP1,@>83D4
        MOV  @RVDP1,0
        MOVB 0,@>8C02
        SWPB 0
        MOVB 0,@>8C02  _________
        MOV  @BRET,11
        RT
*************
SCR2    LWPI WS1
        BL   @SCR20
        MOV  @FLAG,@FLAG  Appel par clavi
        JNE  SCR25
RETOUR  CLR  @FLAG
        CLR  R0
        MOVB R0,@ST
        LWPI >83E0
        B    @>70
SCR25   LWPI >83E0
        BL   @>E
        LWPI WS1
```

```
       CB   @RCLAV,@BEGIN
       JNE  SCR25
       JMP  SCR1
***** GRAPIC MODE **************
SCR10  MOV  11,@BRET
       MOV  @BITMAP,@BITMAP
       JEQ  RETOUR
       CLR  @BITMAP
       MOVB @REG1+2,@>83D4
       LI   1,REG1
SCR12  MOV  *1+,2
       JEQ  SCR11
       MOVB 2,@>8C02
     * SWPB 2            manquante
     * MOVB 2,@>8C02     manquante
       JMP  SCR12
SCR11  CLR  0        ----*
       LI   1,BUFF1
       LI   2,>A00      *
       BLWP @VMBW   _____*
       MOVB @RVDP1,@>83D4
       MOV  @RVDP1,0
       MOVB 0,@>8C02
       SWPB 0
       MOVB 0,@>8C02
       MOV  @BRET,11
       RT
*********
SCR1   LWPI WS1
       BL   @SCR10
       JMP  RETOUR
******************************
FC     EQU  >834A
GN     BLWP @NUMREF
       BLWP @XMLLNK
       DATA >12B8          CFI
       RT
******************************
REG4   DATA 0
DESSIN CI   1,191     R1=Y
       JH   NOD
       CI   0,255     R0=X
       JH   NOD
       MOV  4,@REG4   R4 est utilise
       MOV  R1,R4      par cercle
       SLA  R4,5
       SOC  R1,R4
       ANDI R4,>FF07
       A    R0,R4
       ANDI R0,7
       S    R0,R4
       LI   R3,>8000
       SRC  R3,0
       STWP 1
       CI   R4,>A00
       JGT  DESS1
       SOCB 3,@BUFF2(R4)
       MOVB @BUFF2(R4),3
       JMP  DESS2
```

```
DESS1  MOVB @9(1),@>8C02 lsb of R12
       MOVB R4,@>8C02
       SOCB @>8800,3
DESS2  MOVB @9(1),@>8C02
       ORI  R4,>4000
       MOVB R4,@>8C02
       MOVB 3,@>8C00
*****COLOR
       MOVB @9(1),@>8C02
       AI   R4,>2000
       MOVB R4,@>8C02      R5=COLOR
       MOVB @COULP,@>8C00
       MOV  @REG4,4
NOD    RT
******************************
SAISIE CLR  R0             NOT ARRAY
ELEMENT,R0=1(ARRAY)
       MOV  R11,R12        POUR RT
       LI   R1,1           FIRST ELEMENT
       BL   @GN
       MOV  @FC,R5         COULEUR
       DEC  R5
       SLA  5,12
       MOVB 5,@COULP
       C    R1,R10         COMPTEUR
       JEQ  PLEIN
       INC  R1
       BL   @GN
       MOV  @FC,R6         Y
       C    R1,R10
       JEQ  PLEIN
       INC  R1
       BL   @GN
       MOV  @FC,R7         X
       C    R1,R10
       JEQ  PLEIN
       INC  R1
       BL   @GN
       MOV  @FC,R8        Y' OU RAY CERCLE
       C    R1,R10
       JEQ  PLEIN
       INC  R1
       BL   @GN
       MOV  @FC,R9         X'
PLEIN  B    *R12
*********************
POINT  LWPI WS1
       MOV  @BITMAP,@BITMAP
       JNE  POINT1
       B    @RETOUR
POINT1 LI   R10,3          3 PARAMETRES
       BL   @SAISIE
       MOV  R6,R1
       MOV  R7,R0
       BL   @DESSIN
       B    @RETOUR
********************
ORIGIN LWPI WS1
       MOVB @>8312,10
```

```
        SRL  10,8
        CI   10,2
        JEQ  OR1
OR2     B    @RETOUR
OR1     CLR  0
        LI   1,1
        BL   @GN
        MOV  @FC,@Y1
        INC  1
        BL   @GN
        MOV  @FC,@X1
        JMP  OR2
*********************
Y1      BSS  2
X1      BSS  2
LIGNE   LWPI WS1
        MOV  @BITMAP,@BITMAP
        JNE  LIGNE1
        B    @RETOUR
LIGNE1  MOVB @>8312,10
        SRL  10,8
        BL   @SAISIE
        CI   10,3
        JEQ  LIGNE2
        CI   10,5
        JNE  LIG6
        JMP  LIGNE3
LIGNE2  MOV  6,8
        MOV  7,9
        MOV  @Y1,6
        MOV  @X1,7
LIGNE3  MOV  R8,@Y1
        MOV  R9,@X1
        CLR  R12
        LI   R14,1
        LI   R15,1
        S    R7,R9
        JLT  $+4
        JMP  $+6
        NEG  R14
        NEG  R9
        MOV  R9,R9
        JNE  $+4
        SETO R12
        S    R6,R8
        JLT  $+4
        JMP  LIG2
        NEG  R15
        NEG  R8
LIG2    MOV  R7,R0
        MOV  R6,R1
        BL   @DESSIN
        C    R7,@X1
        JNE  LIG4
        CLR  14
        C    R6,@Y1
        JNE  LIG4
LIG6    B    @RETOUR
LIG4    MOV  R12,R12
        JLT  LIG3
        A    R14,R7
        S    R8,R12
        JLT  LIG5
        JMP  LIG2
LIG5    A    15,6
        A    9,12
        JMP  LIG2
LIG3    A    15,6
        A    9,12
        JLT  LIG2
        A    14,7
        S    8,12
        JMP  LIG2
******************************
RCLAV   EQU  >8375
BEGIN   BYTE 14
PROC    BYTE 12
***
INTER   CB   @RCLAV,@BEGIN
        JEQ  INT1
        CB   @RCLAV,@PROC
        JEQ  INT2
RET     RT
INT1    MOV  @BITMAP,@BITMAP
        JEQ  RET
        B    @SCR1
INT2    MOV  @BITMAP,@BITMAP
        JNE  RET
        SETO @FLAG    attendre l'appui
        B    @SCR2    sur begin
******
DEL     DATA 50
FIL3    DATA 0
FIL1    DATA >AA3F,>FF11,>100
FILES1  INC  @FIL3
        C    @FIL3,@DEL
        JNE  RET
        BLWP @FIL2
        MOV  @WS1,@>83C4
        RT
FIL2    DATA WS1,FIL2+4
        LI   0,>3BE4
        LI   1,FIL1
        LI   2,5
        BLWP @VMBW
        LI   0,>3BE3
        MOV  0,@>8370
        LI   0,INTER
        RTWP
******
DAA     BYTE >AA
D20     BYTE >20
D2E     BYTE >2E  "."
        EVEN
***
DSRLNK  DATA >209A,DSRLNK+4
        MOV  *14+,5
        SZCB @D20,15
```

```
       MOV  @>8356,0
       MOV  0,9
       AI   9,>FFF8
       BLWP @VSBR
       MOVB 1,3
       SRL  3,8
       SETO 4
       LI   2,>208C
S001   INC  0
       INC  4
       C    4,3
       JEQ  S000
       BLWP @VSBR
       MOVB 1,*2+
       CB   1,@D2E
       JNE  S001
S000   MOV  4,4
       JEQ  S002
       CI   4,7
       JGT  S002
       CLR  @>83D0
       MOV  4,@>8354
       MOV  4,@>2036
       INC  4
       A    4,@>8356
       MOV  @>8356,@>2038
       LWPI >83E0
       CLR  1
       LI   12,>F00
S005   MOV  12,12
       JEQ  S003
       SBZ  0
S003   AI   12,>100
       CLR  @>83D0
       CI   12,>2000
       JEQ  S004
       MOV  12,@>83D0
       SBO  0
       LI   2,>4000
       CB   *2,@DAA
       JNE  S005
       A    @>20A4,2
       JMP  S006
S008   MOV  @>83D2,2
       SBO  0
S006   MOV  *2,2
       JEQ  S005
       MOV  2,@>83D2
       INCT 2
       MOV  *2+,9
       MOVB @>8355,5
       JEQ  S007
       CB   5,*2+
       JNE  S008
       SRL  5,8
       LI   6,>208C
S009   CB   *6+,*2+
       JNE  S008
       DEC  5
```

```
       JNE  S009
S007   INC  1
       MOV  1,@>203A
       MOV  9,@>2034
       MOV  12,@>2032
       BL   *9
       JMP  S008
       SBZ  0
       LWPI >209A
       MOV  9,0
       BLWP @VSBR
       SRL  1,13
       JNE  S010
       RTWP
S004   LWPI >209A
S002   CLR  1
S010   SWPB 1
       MOVB 1,*13
       SOCB @D20,15
       RTWP
**************
PAB    EQU  >3800
PLONG  EQU  >8356
FLAG1  DATA 0      1>sauve 2>charge
FLAG2  DATA 0      1 si appel sauv/
BRET   DATA 0       charge en SCR1
SAVE   DATA >600
LOAD   DATA >500
DSK    DATA 0,0,0,>1800,0
TDSK   TEXT '
       EVEN
LNOM   EQU  DSK+9
****
SAIS#  DATA 0      Return bufer
SAIS   MOV  11,@SAIS#
       CLR  @FLAG2
       MOV  @BITMAP,@BITMAP
       JNE  SAIS1
       SETO @FLAG2  Appel en SCR1
       BL   @SCR20
SAIS1  LI   0,>F00
       MOVB 0,@LNOM
       CLR  0
       LI   1,1
       LI   2,LNOM
       BLWP @STRREF
       MOV  @SAIS#,11
       RT
*******
SAUVE  LWPI WS1
       SETO @FLAG1
       BL   @SAIS
       MOV  @SAVE,@DSK
       CLR  0          ----*
       LI   1,BUFF2        *
       LI   2,>A00         *
       BLWP @VMBW      ____*
       JMP  SACH
CHARGE LWPI WS1
```

```
       CLR  @FLAG1    0=charge
       BL   @SAIS
       MOV  @LOAD,@DSK
       LI   0,>2000      ---*Couleur blanc
       LI   1,>F000         *sur transpar.
       LI   2,>1800         *
       BL   @CL1         ___*
SACH   LI   0,PAB
       LI   1,DSK
       LI   2,25
       BLWP @VMBW
       LI   0,PAB+9
       MOV  0,@PLONG
       CLR  1
       MOVB 1,@>837C
       BLWP @DSRLNK
       DATA 8
       JEQ  ERR
       MOV  @FLAG1,@FLAG1     1=sauve
       JNE  SACH1
       CLR  0         ----*
       LI   1,BUFF2       *
       LI   2,>A00        *
       BLWP @VMBR    _____*
       CLR  0         --------*
SACH3  MOVB @WS1+1,@>8C02   *
       MOVB 0,@>8C02        *
       MOVB @>8800,1        *
       JNE  SACH4           *
       MOVB @WS1+1,@>8C02   *
       ORI  0,>6000         *
       MOVB 0,@>8C02        *
       MOVB 1,@>8C00        *
       ANDI 0,>1FFF         *
SACH4  INC  0               *
       CI   0,>1800         *
       JNE  SACH3   ________*
SACH1  MOV  @FLAG2,@FLAG2  Si appel en
       JEQ  SACH2          SCR1
       B    @SCR1
SACH2  B    @RETOUR
***
ERR    MOV  @BITMAP,@BITMAP
       JEQ  ERR1
       BL   @SCR10
ERR1   LI   0,>2D00   I/O ERR CODE
       MOV  0,@>8322
       LWPI >83E0
       SOCB @>11B,@>837C
       LI   1,PAB
       MOV  1,@>831C
       B    @>70
******
CERC1  MOV  2,@WS1
       MOV  4,@WS1+2
       MPY  2,2
       MPY  4,4
       A    3,5
       MOV  @WS1,2
       MOV  @WS1+2,4
       C    5,10
       RT
CERC   DATA WS2,CERC+4
       CLR  2
       MOV  @16(13),10    R
       MOV  10,4
       DEC  4
       CI   10,1
       JGT  B1
       JLT  FIN
       MOV  @14(13),0
       MOV  @12(13),1
       BL   @DESSIN
       JMP  FIN
B1     MPY  10,10
       MOV  11,10
       LI   12,CERC1
B      MOV  @14(13),0
       MOV  @12(13),1
       A    2,0
       S    4,1
       BL   @DESSIN
       MOV  @14(13),0
       MOV  @12(13),1
       A    2,0
       A    4,1
       BL   @DESSIN
       MOV  @14(13),0
       MOV  @12(13),1
       S    2,0
       A    4,1
       BL   @DESSIN
       MOV  @14(13),0
       MOV  @12(13),1
       S    2,0
       S    4,1
       BL   @DESSIN
       MOV  4,4
       JEQ  FIN
       INC  2
       BL   *12
       JHE  Y
       JMP  B
Y      DEC  4
       BL   *12
       JHE  YY
       JMP  B
YY     DEC  2
       JMP  B
FIN    RTWP
CERCLE LWPI WS1
       LI   10,4
       BL   @SAISIE
       BLWP @CERC
       B    @RETOUR
       AORG >83C4  USER INTERRUPT
ROUTINE
       DATA FILES1
       END
```

49

Programme de démonstration

```
1 CALL LOAD(-31890,56,2):
  : CALL LOAD(-31964,56,
  0)
2 Z1,V7,U7,K7,CC=DD !Pour
   un programme en TI-ba
   sic,supprimer les lign
   es 5-20 ainsi que la 2
   e partie de 90,et remp
   lacer la ligne 620 par
   621
3 GOTO 5 :: CALL PEEK ::
  CALL CLEAR :: CALL LOA
  D :: CALL LINK :: XF,Y
  F,Z,UU,AD,AD1,A,B,REP,
  N,E1,E2,E3,XA,YA,XB,YB
  ,XC,YC,XD,YD,I,XP,YP,X
  Q,YQ,X,Y,I1,I2,J,G,X1,
  Y1,X7,Y7,DI,AA,BB,M7,M
  8=M9
4 !@P-
5 CALL PEEK(-31866,A,B)::
   AD=A*256+B-65535+4652
   !-31866=END OF FREE S
   PACE. 4652=NUMERIC VAL
   UES TABLES
10 CALL PEEK(AD,A,B):: AD
   =AD+4 :: IF A*256+B<>6
   20 THEN 10
20 CALL PEEK(AD-2,A,B)::
   AD=A*256+B-65536
70 X1=0 :: Y1=0
80 CALL CLEAR :: DISPLAY
   AT(5,3)BEEP:"CHOISISSE
   Z UN NOMBRE ?         0
   ,1,2,3,4,5,6,7,8 :" ::
    ACCEPT AT(9,3)VALIDAT
   E(DIGIT):REP
85 CALL LINK("CLEAR"):: C
   ALL LINK("SCR2")
90 ON REP+1 GOSUB 100,101
   ,102,103,104,105,106,1
   07,108 :: AD1=5000+REP
   *100 :: A=INT(AD1/256)
   :: B=AD1-A*256 :: CALL
    LOAD(AD+2,A,B):: GOTO
    200
100 N=30 :: PA=3 :: E1=2
   :: E2=1 :: XA=128 :: Y
   A=5 :: XB=250 :: YB=55
    :: XC=110 :: YC=150 :
   : XD=10 :: YD=105 :: R
   ETURN
101 N=20 :: PA=3 :: E1=2
   :: E2=0 :: XA=85 :: YA
   =0 :: XB=255 :: YB=65
   :: XC=170 :: YC=191 ::
    XD=0 :: YD=125 :: RET
   URN
102 N=45 :: PA=2 :: E1=2
   :: E2=0 :: XA=128 :: Y
   A=0 :: XB=254 :: YB=96
    :: XC=128 :: YC=190 :
   : XD=0 :: YD=96 :: RET
   URN
103 N=30 :: PA=3 :: E1=2
   :: E2=0 :: XA=80 :: YA
   =0 :: XB=220 :: YB=45
   :: XC=160 :: YC=130 ::
    XD=10 :: YD=69 :: RET
   URN
104 N=31 :: PA=2 :: E1=2
   :: E2=0 :: XA=128 :: Y
   A=15 :: XB=255 :: YB=8
   0 :: XC=128 :: YC=160
   :: XD=0 :: YD=80 :: RE
   TURN
105 N=66 :: PA=2 :: E1=1
   :: E2=0 :: XA=128 :: Y
   A=0 :: XB=255 :: YB=85
    :: XC=128 :: YC=170 :
   : XD=0 :: YD=85 :: RET
   URN
106 N=54 :: PA=1 :: E1=1
   :: E2=0 :: XA=80 :: YA
   =20 :: XB=255 :: YB=20
    :: XC=175 :: YC=111 :
   : XD=0 :: YD=111 :: RE
   TURN !N=36
107 N=40 :: PA=2 :: E1=2
   :: E2=0 :: XA=100 :: Y
   A=10 :: XB=255 :: YB=9
   0 :: XC=155 :: YC=170
   :: XD=0 :: YD=90 :: RE
   TURN
108 N=30 :: PA=2 :: E1=2
   :: E2=0 :: XA=85 :: YA
   =0 :: XB=255 :: YB=30
   :: XC=170 :: YC=90 ::
   XD=0 :: YD=60 :: RETUR
   N
199 !@P+
200 M=255/PA :: DIM MA(25
   5),MI(255)
201 !@P-
300 FOR I=0 TO M :: MA(I)
   =-500 :: MI(I)=500 ::
   NEXT I
500 FOR I=0 TO N
510 XP=(I*XD+(N-I)*XA)/N
520 YP=(I*YD+(N-I)*YA)/N
530 XQ=(I*XC+(N-I)*XB)/N
540 YQ=(I*YC+(N-I)*YB)/N
550 IF E3=1 THEN Y=I/N EL
   SE X=I/N
560 I1=INT(XP/PA):: I2=IN
   T(XQ/PA):: G=SGN(I2-I1
   )
600 FOR J=I1 TO I2 STEP G
610 IF E3=1 THEN X=(J-I1)
   /(I2-I1)ELSE Y=(J-I1)/
   (I2-I1)
620 GOSUB 5100
630 XF=J*PA
640 YF=((J-I1)*YQ+(I2-J)*
   YP)/(I2-I1)+Z
700 IF J=I1 THEN CALL LIN
   K("ORIGIN",191-YF,XF)
710 IF E2=1 THEN 760
720 IF YF>MI(J)AND YF<MA(
   J)THEN CALL LINK("ORIG
   IN",191-YF,XF):: GOTO
   800
740 IF YF>MA(J)THEN MA(J)
   =YF
750 IF YF<MI(J)THEN MI(J)
   =YF
760 CALL LINK("LIGNE",16,
   191-YF,XF)
800 NEXT J
900 NEXT I
1000 IF E1=1 THEN 9999
1010 E3=1 :: E1=1
1020 UU=XD :: XD=XB :: XB
   =UU
1030 UU=YD :: YD=YB :: YB
   =UU
1050 GOTO 300
5000 Z=-40*SIN(2*PI*Y)*SI
   N(PI*X):: RETURN
5100 X7=2*X-1 :: Y7=2*Y-1
    :: DI=X7*X7+Y7*Y7 ::
   IF DI<1 THEN Z=SQR(1-D
   I)ELSE Z=0
5110 Z=60*Z :: RETURN
5200 Z=50*SIN(2*PI*Y)*SIN
   (3*PI*X):: RETURN
5300 X7=2*X-1 :: Y7=2*Y-1
5310 AA=SQR(1-X7*X7)*(ABS
   (Y7)+.5):: BB=.7*SQR(1
   -Y7*Y7)*(ABS(X7)+.5)
5320 IF BB>AA THEN AA=BB
5330 Z=60*AA :: RETURN
5400 DI=8*SQR((X-.5)*(X-.
   5)+(Y-.5)*(Y-.5)):: Z=
   40*COS(DI):: RETURN
5500 DI=16*((X-.5)*(X-.5)
   +(Y-.5)*(Y-.5))
5510 Z=100*COS(4*DI)*EXP(
   -DI):: RETURN
5600 X7=X :: Y7=Y :: K7=0
5610 K7=K7+1 :: IF K7>5 T
   HEN Z=0 :: RETURN
5620 U7=INT(3*X7):: X7=3*
   X7-U7
5630 V7=INT(3*Y7):: Y7=3*
   Y7-V7
5640 IF U7<>1 OR V7<>1 TH
   EN 5610
5650 Z1=.5-ABS(X7-.5):: Z
   =.5-ABS(Y7-.5):: IF Z1
   <Z THEN Z=Z1
5660 Z=(500/3^K7)*Z :: RE
   TURN
5700 X7=2*X-INT(2*X):: Y7
   =2*Y-INT(2*Y):: AA=1.5
   -6*ABS(X7-.5):: BB=1.5
   -6*ABS(Y7-.5):: IF BB<
   AA THEN AA=BB
5710 CC=.5-2*ABS(X7-.5)::
    DD=.25-ABS(Y7-.5):: I
   F CC>AA THEN AA=CC
5720 IF DD>AA THEN AA=DD
5730 Z=40*AA :: RETURN
5800 X7=2*ABS(X-.5):: Y7=
   2*ABS(Y-.5):: M7=X7+Y7
    :: M8=INT(3*M7):: M9=
   3*M7-M8 :: Z=0
5810 IF M9>.8 THEN Z=5*(M
   9-.8)
5820 Z=136-24*(Z+M8):: RE
   TURN
9999 GOTO 9999
```

Alligator

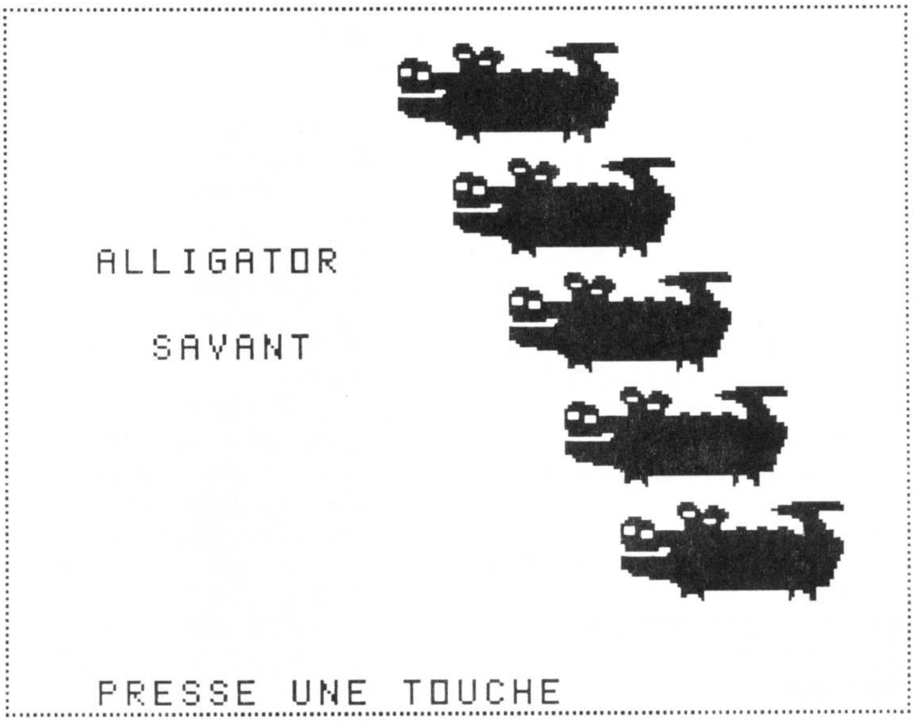
Bernard Legrand

Ce jeu éducatif est très largement inspiré du célèbre module "Alligator Mix" de Texas Instruments.

Différents menus vous offrent le choix entre plusieurs options de jeu. Le programme vous guide pas à pas, ce qui fait qu'un mode d'emploi serait ici superflu.

Le but est de faire manger la pomme mobile par le crocodile lorsque la réponse contenue par le fruit en correcte. Chaque pression sur la touche 'espace' ouvre ou ferme la gueule de l'animal.

99



Programme "Alligator savant"

Basic Étendu

```
100 REM
110 REM
120 REM *************
130 REM **ALLIGATOR**
140 REM **  SAVANT **
150 REM *************
160 REM
170 REM
180 DIM OP$(3),STAT(4,2),
   O$(4),S$(4)
190 CALL CLEAR :: CALL MA
   GNIFY(3):: RANDOMIZE :
   : FIN,COR,FAUX=0 :: GO
   TO 1190
200 REM
210 REM DEBUT DE LA BOUCL
   E
220 REM PRINCIPALE
230 REM
240 CALL SPRITE(#8,140,1,
   FJ,35,#7,136,1,FJ,21,#
   6,124,1,F2J,35,#5,120,
   1,F2J,18)
250 CALL MOTION(#5,0,SPD,
   #6,0,SPD,#8,0,SPD,#7,0
   ,SPD)
260 CALL COLOR(#5,2,#6,2,
   #7,16,#8,16):: CALL CO
   LOR(8,2,16):: FLG=0
270 CALL KEY(0,K,S):: IF
   S=0 THEN 300
280 IF FLG=0 THEN FLG=1 E
   LSE FLG=0
290 CW=96+20*FLG :: CALL
   HCHAR(FA,FB,CW):: CALL
    HCHAR(FA,FD,CW+2):: C
   ALL HCHAR(FC,FB,CW+1):
   : CALL HCHAR(FC,FD,CW+
   3)
300 CALL COINC(#8,FJ,GJ,6
   ,COIN):: IF COIN=0 THE
   N 270
310 CALL DELSPRITE(ALL)::
    CALL SPRITE(#7,136,16
   ,FJ,GJ-12,#8,140,16,FJ
   ,GJ+4,#5,120,2,F2J,GJ-
   12,#6,124,2,F2J,GJ+4)
320 IF (FLG=0 AND X=1)OR(
   FLG=1 AND X=0)THEN 420
330 REM
340 REM ACTION FAUSSE
350 REM
360 CALL SOUND(200,SOL,0,
   SOL2,4):: CALL SOUND(2
   00,MI,0,MI2,4):: CALL
   SOUND(200,DO1,0,DO12,4
   )
370 FAUX=FAUX+1 :: IF J=5
    OR I=0 THEN 380 ELSE
   CALL AF(I,J+1,0):: I=I
   -1 :: GA=0
380 A$=STR$(FAUX):: DISPL
   AY AT(24,21+6-LEN(A$))
   SIZE(LEN(A$)):A$ :: ST
   AT(OPE,2)=STAT(OPE,2)+
   1 :: GOTO 540
390 REM
400 REM ACTION JUSTE
410 REM
420 CALL SOUND(200,DO1,0,
   DO12,4):: CALL SOUND(2
   00,DO2,0,DO22,4):: CAL
   L SOUND(200,SOL,0,SOL2
   ,4)
430 COR=COR+1 :: STAT(OPE
   ,1)=STAT(OPE,1)+1 :: I
   F J=5 THEN 460
440 GA=GA+1 :: IF GA<2 TH
   EN 460
```

```
450 GA=1 :: I=I+1 :: IF I
   <9 THEN CALL AF(I,J+1,
   1)
460 A$=STR$(COR):: DISPLA
   Y AT(24,11+3-LEN(A$))S
   IZE(LEN(A$)):A$
470 IF I<>8 THEN 540 ELSE
    J=J+1
480 I=0 :: GA=-1 :: CALL
   CHAR(88,"0"):: CALL CH
   AR(89,"0")
490 CALL HCHAR((5-J)*4+4,
   30-2*J+(I-1)*2,88):: C
   ALL HCHAR(24-4*J,29-2*
   J+2*I,89)
500 IF J=1 THEN 520 ELSE
   CALL AF(2,J-1,1):: CAL
   L AF(3,J-1,1)
510 CALL AF(5,J-1,1)
520 FJ=175-32*J :: F2J=17
   9-32*J :: GJ=180-16*J
530 FA=22-4*J :: FB=25-2*
   J :: FC=FA+1 :: FD=FB+
   1
540 CALL DELSPRITE(#5,#6,
   #7,#8):: CALL CHAR(88,
   "0"):: CALL CHAR(89,"0
   "):: CALL AF(5,J,1)
550 IF COR+FAUX=DUR THEN
   1040
560 REM
570 REM CHOIX DE L'OPERAT
   ION
580 REM SUIVANT L'OPTION
590 REM CHOISIE
600 REM
610 AMA=99 :: ON OPT GOTO
    660,830,890,700
620 REM
630 REM ADDITION ET
640 REM SOUSTRACTION
650 REM
660 OPE=INT(RND*2)+1 :: O
   N OPE GOTO 740,780
670 REM
680 REM LES 4 OPERATIONS
690 REM
700 OPE=INT(RND*4)+1 :: O
   N OPE GOTO 740,780,830
   ,890
710 REM
720 REM ADDITION
730 REM
740 B=INT(RND*10):: A=INT
   (RND*21):: GOTO 790
750 REM
760 REM SOUSTRACTION
770 REM
780 B=INT(RND*10):: IF SO
   =1 THEN A=INT(RND*10)+
   B :: AMA=9 ELSE A=INT(
   RND*21)+B
790 CALL CHAR(124,O$(OPE)
   ):: RESU=(A+B)*(2-OPE)
   +(A-B)*(OPE-1):: CALL
   AFC(A,B,120,124,T$()):
   : GOTO 950
800 REM
810 REM MULTIPLICATION
820 REM
830 OPE=3 :: IF MUL<>0 TH
   EN B=MUL :: GOTO 850
840 B=INT(RND*9)+1
850 A=INT(RND*12):: RESU=
   A*B :: CALL AFC(A,B,12
   4,118,T$()):: CALL CHA
   R(122,O$(3)):: GOTO 95
   0
860 REM
870 REM DIVISION
880 REM
890 OPE=4 :: B=INT(RND*9)
   +1 :: RESU=INT(RND*12)
   :: A=B*RESU :: CALL CH
   AR(124,O$(4))
900 CALL AFC(A,B,120,124,
   T$())
910 REM
920 REM AFFICHAGE JUSTE O
   U
930 REM             FAUX ?
940 REM
950 X=INT(RND*3):: IF X<2
    THEN X=0 :: GOTO 980
960 X=1 :: E1=INT(OPE/4)+
   (1-INT(OPE/4))*(4*(1-I
   NT(DIF/4))+2*INT(DIF/4
   )):: A=MAX(0,RESU-E1):
   : B=MIN(RESU+E1,AMA)
970 EC=INT((B-A+1)*RND)+A
    :: IF EC=RESU THEN 97
   0 ELSE RESU=EC
980 IF RESU>=10 THEN CALL
    CHAR(88,T$(INT(RESU/1
   0)+.1))ELSE CALL CHAR(
   88,"0")
990 CALL CHAR(89,T$(INT(R
   ESU-INT(RESU/10)*10+.1
   ))):: GOTO 240
1000 GOTO 1040
1010 REM
1020 REM AFFICHAGE DES RE
   SULTATS
1030 REM
1040 CALL DELSPRITE(ALL):
   : CALL CLEAR :: FIN=1
1050 RE$(1)="JUSTES ," ::
    RE$(2)="FAUSSES"
1060 DISPLAY AT(1,7):"TES
    RESULTATS" :: ON OPT
   GOTO 1070,1090,1100,11
   10
1070 DISPLAY AT(6,2):"ADD
   ITION ET SOUSTRACTION"
    :: CALL AFR(10,"+",ST
   AT(1,1),STAT(1,2))
1080 CALL AFR(14,"-",STAT
   (2,1),STAT(2,2)):: GOT
   O 1130
1090 DISPLAY AT(6,2):"MUL
   TIPLICATION" :: CALL A
   FR(10,"*",STAT(3,1),ST
   AT(3,2)):: GOTO 1130
1100 DISPLAY AT(6,2):"DIV
   ISION" :: CALL AFR(10,
   "/",STAT(4,1),STAT(4,2
   )):: GOTO 1130
1110 DISPLAY AT(6,2):"LES
    4 OPERATIONS   MELANG
   EES" :: CALL AFR(10,"+
   ",STAT(1,1),STAT(1,2))
   :: CALL AFR(12,"-",STA
   T(2,1),STAT(2,2))
1120 CALL AFR(14,"*",STAT
   (3,1),STAT(3,2)):: CAL
   L AFR(16,"/",STAT(4,1)
   ,STAT(4,2)):: GOTO 113
   0
1130 CHAN=0 :: DISPLAY AT
   (24,2):"PRESSE UNE TOU
   CHE"
1140 CALL KEY(0,K,S):: IF
    S<>0 THEN CALL CLEAR
   :: GOTO 1550
1150 CHAN=CHAN+1 :: IF S=
   0 AND CHAN=500 THEN CA
   LL CLEAR :: GOTO 1550
   ELSE 1140
1160 REM
1170 REM DEFINITION DES C
   ARACTERES
1180 REM
1190 CALL CHAR(96,"000000
   003C7FFF8F8CFCFFFF7F7F
   00FF000000000080C0C07F
   7FFFFFFFE307FF")
1200 DO1=110*2^(3/12):: D
   O12=2*DO1 :: DO2=110*2
   ^(15/12):: DO22=2*DO2
   :: SOL=110*2^(10/12)::
    SOL2=2*SOL
1210 MI=110*2^(7/12):: MI
   2=2*MI
1220 CALL CHAR(100,"78FCF
   EC77F3F3F3F0070FCFEFE1
   CF8F800000000000000710
   0071F07000000C7")
1230 CALL CHAR(104,"7FFFF
   FFF7F1F0F0FFEFFFF8080C
   0E0E0FFFFFFFFFFFFFFFFC
   FFFFFFFFFFFFFF")
1240 CALL CHAR(108,"F8F8F
   EFFFFFFFFFFEFFFFFF3F1F0
   00000FFFFFFFFFFF0300F
   EFCFCF8F8F8F0F0")
1250 CALL CHAR(112,"7F7F6
   30100000000FF000000000
   00000FFCFC38300000000"
   )
1260 CALL CHAR(116,"3C7EC
   7C4FC3F3F0707010000000
   000CC0000808080E0F8FEF
   FFF7F1F070100CC")
1270 OP$(0)="ADDITION ET
   SOUSTRACTION" :: OP$(1
   )="MULTIPLICATION" ::
   OP$(2)="  LA DIVISION"
1280 OP$(3)="   LES 4 OPE
```

```
RATIONS"
1290 FOR I=0 TO 9 :: CALL
   CHARPAT(I+48,T$(I)):: 
   NEXT I
1300 CALL CHARPAT(43,O$(1
   ),45,O$(2),42,O$(3),47
   ,O$(4))
1310 CALL CHAR(140,"C31F7
   F"&RPT$("F",25)&"8C0F8
   FCFE"&RPT$("F",15)&"EF
   CF88000")
1320 CALL CHAR(136,"01073
   F7F"&RPT$("F",14)&"7F1
   F070000C0F9FC"&RPT$("F
   ",24)&"1F")
1330 CALL CHAR(124,RPT$("
   0",64)):: CALL CHAR(12
   0,RPT$("0",64))
1340 CALL CHARPAT(88,C$):
   : CALL CHAR(87,C$)
1350 REM
1360 REM DESSIN INTRODUCT
   IF
1370 REM
1380 CALL COLOR(9,1,1,10,
   1,1,11,1,1):: FIN=0
1390 FOR J=1 TO 5 :: FOR
   I=1 TO 8 :: CALL AF(I,
   J,1):: NEXT I :: NEXT
   J
1400 DISPLAY AT(9,2)SIZE(
   9):"ALLIGATOR" :: DISP
   LAY AT(12,4)SIZE(6):"S
   AVANT" :: DISPLAY AT(2
   4,2):"PRESSE UNE TOUCH
   E"
1410 CALL COLOR(9,2,1,10,
   2,1,11,2,1):: CHAN=0 :
   : CALL SPRITE(#4,96,2,
   137,177):: CALL AF(5,1
   ,0)
1420 CALL SPRITE(#8,140,1
   ,143,35,#7,136,1,143,2
   1)
1430 CALL MOTION(#8,0,9,#
   7,0,9):: CALL COLOR(#8
   ,16,#7,16):: DC=161 ::
   PAT,PA=0
1440 CALL KEY(0,K,S):: IF
    S<>0 OR CHAN=10 THEN
   CALL PATTERN(#4,96)::
   CALL AF(5,1,1):: GOTO
   1500
1450 PAT=PAT+1 :: IF PAT<
   10 THEN 1470 ELSE PAT=
   0 :: CALL PATTERN(#4,9
   6*(1-PA)+116*PA)
1460 IF PA=0 THEN PA=1 EL
   SE PA=0
1470 CALL COINC(#8,143,DC
   ,4,COIN):: IF COIN=0 T
   HEN 1440
1480 IF DC=161 THEN DC=35
    :: CALL MOTION(#8,0,-
   9,#7,0,-9):: CHAN=CHAN
   +1 :: GOTO 1440
1490 DC=161 :: CALL MOTIO
   N(#8,0,9,#7,0,9):: CHA
   N=CHAN+1 :: GOTO 1440
1500 CALL DELSPRITE(ALL):
   : CALL COLOR(9,1,1,10,
   1,1,11,1,1):: DISPLAY
   AT(9,2)SIZE(9):" " ::
   DISPLAY AT(12,4)SIZE(6
   ):" "
1510 DISPLAY AT(24,1):" "
1520 REM
1530 REM CHOIX DES OPTION
   S DE JEU
1540 REM
1550 DISPLAY AT(1,2)BEEP:
   "CHOIW DES OPTIONS DE
   JEUW" :: DISPLAY AT(4,
   1)SIZE(12):"1 ADDITION
    &" :: DISPLAY AT(6,1)
   SIZE(14):"  SOUSTRACTI
   ON"
1560 DISPLAY AT(9,1)SIZE(
   16):"2 MULTIPLICATION"
    :: DISPLAY AT(12,1)SI
   ZE(16):"3 DIVISION"
1570 DISPLAY AT(15,1)SIZE
   (18):"4 LES 4 OPERATIO
   NS" :: DISPLAY AT(17,1
   )SIZE(11):"
  MELANGEES"
1580 DISPLAY AT(20,1)SIZE
   (12):"5 FIN DU JEU"
1590 CHAN=0 :: I=17 :: OP
   T=1
1600 DISPLAY AT(24,3)SIZE
   (11):"TON CHOIW :" ::
   CALL HCHAR(24,I,30+INT
   (CHAN-2*INT(CHAN/2))*(
   18+OPT)):: CALL KEY(0,
   K,S):: CHAN=CHAN+.5
1610 IF S=0 AND CHAN=100
   THEN 1620 ELSE 1660
1620 FOR I=3 TO 8 :: CALL
    COLOR(I,1,1):: NEXT I
    :: DISPLAY AT(1,2):"
   " :: DISPLAY AT(4,1)SI
   ZE(12):" " :: DISPLAY
   AT(6,1)SIZE(14):" "
1630 DISPLAY AT(9,1)SIZE(
   16):" " :: DISPLAY AT(
   12,1)SIZE(16):" " :: D
   ISPLAY AT(20,1)SIZE(12
   ):" " :: DISPLAY AT(24
   ,3)SIZE(13):" "
1640 DISPLAY AT(15,1)SIZE
   (18):" " :: DISPLAY AT
   (17,1)SIZE(12):" " ::
   FOR I=3 TO 8 :: CALL C
   OLOR(I,2,1):: NEXT I
1650 IF FIN=0 THEN 1400 E
   LSE 1380
1660 IF S=0 THEN 1600 ELS
   E IF K<49 OR K>53 THEN
    CHAN=0 :: IF K=13 THE
   N 1680 ELSE 1600
1670 CALL HCHAR(24,17,K):
   : OPT=K-48 :: GOTO 160
   0
1680 IF OPT=5 THEN CALL C
   LEAR :: STOP
1690 COR,FAUX=0 :: CALL C
   LEAR :: DISPLAY AT(5,1
   ):"D'ACCORD POUR": :
   OP$(OPT-1):: ON OPT GO
   TO 1700,1720,1750,1750
1700 DISPLAY AT(15,1):"PO
   UR LES SOUSTRACTIONS":
   :"VEUW TU LES RESULTA
   TS": :"SUR 1 OU 2 CHIF
   FRES   2" :: ACCEPT AT
   (19,23)SIZE(-1)VALIDAT
   E("12"):A$
1710 SO=ASC(A$)-48 :: GOT
   O 1760
1720 DISPLAY AT(15,1):"VE
   UW TU UNE SEULE TABLE
             (O/N) ? N" :: A
   CCEPT AT(16,12)SIZE(-1
   )VALIDATE("ON"):A$
1730 IF A$="N" THEN MUL=0
    :: GOTO 1760
1740 DISPLAY AT(18,1)BEEP
   :"LAQUELLE (1-9) ? 9"
   :: ACCEPT AT(18,18)SIZ
   E(-1):MUL :: IF MUL<1
   OR MUL>9 THEN 1740
1750 FOR I=1 TO 200 :: NE
   XT I
1760 CALL CLEAR :: DISPLA
   Y AT(3,1):"NIVEAU DE D
   IFFICULTE": :"      (1-
   5) ? 3" :: ACCEPT AT(5
   ,14)SIZE(-1)BEEP:DIF
1770 IF DIF<1 OR DIF>5 TH
   EN 1760 ELSE SPD=DIF*3
   +5
1780 DISPLAY AT(8,1):"NOM
   BRE D'ESSAIS":"  1 POU
   R  60":"  2 POUR 120":
    :"  TON CHOIW (1-2) ?
   1"
1790 ACCEPT AT(12,21)SIZE
   (-1)BEEP:DUR :: IF DUR
   <1 OR DUR>2 THEN 1790
   ELSE DUR=DUR*60
1800 FOR I=1 TO 100 :: NE
   XT I
1810 CALL CLEAR :: FOR I=
   1 TO 8 :: CALL AF(I,1,
   1):: NEXT I :: CALL CO
   LOR(9,2,1,10,2,1,11,2,
   1)
1820 FOR I=1 TO 4 :: FOR
   J=1 TO 2 :: STAT(I,J)=
   0 :: NEXT J :: NEXT I
1830 J=1 :: FOR I=3 TO 7
   :: CALL COLOR(I,2,1)::
    NEXT I :: DISPLAY AT(
   24,2):"CORRECT  000
    FAUW  000"
1840 REM
1850 REM SUBROUTINE D'AFF
```

```
   ICHAGE
1860 REM D'UN ELEMENT DU
   CROCODILE,SI K=1   OU
   D'EFFACEMENT ,SI K=0
1870 REM
1880 GOTO 480
1890 SUB AF(I,J,K)
1900 ON I GOTO 1910,1920,
   1930,1940,1950,1960,19
   70,1980
1910 A=77*K+32 :: B=78*K+
   32 :: C=32 :: D=32 ::
   GOTO 1990
1920 A=74*K+32 :: B=74*K+
   32 :: C=80*K+32 :: D=8
   1*K+32 :: GOTO 1990
1930 A=74*K+32 :: B=74*K+
   32 :: C=81*K+32 :: D=8
   1*K+32 :: GOTO 1990
1940 A=74*K+32 :: B=79*K+
   32 :: C=82*K+32 :: D=3
   2 :: GOTO 1990
1950 A=64*K+32 :: B=66*K+
   32 :: C=65*K+32 :: D=6
   7*K+32 :: GOTO 1990
1960 A=68*K+32 :: B=69*K+
   32 :: C=74*K+32 :: D=7
   4*K+32 :: GOTO 1990
1970 A=70*K+32 :: B=71*K+
   32 :: C=74*K+32 :: D=7
   4*K+32 :: GOTO 1990
1980 A=72*K+32 :: B=73*K+
   32 :: C=75*K+32 :: D=7
   6*K+32 :: GOTO 1990
1990 CALL HCHAR((5-J)*4+2
   +2*INT((8-I)/4),15+(5-
   J)*2+(I-1-4*INT((I-1)/
   4))*2,A)
2000 CALL HCHAR((5-J)*4+2
   +2*INT((8-I)/4),16+(5-
   J)*2+(I-1-4*INT((I-1)/
   4))*2,B)
2010 CALL HCHAR((5-J)*4+3
   +2*INT((8-I)/4),15+(5-
   J)*2+(I-1-4*INT((I-1)/
   4))*2,C)
2020 CALL HCHAR((5-J)*4+3
   +2*INT((8-I)/4),16+(5-
   J)*2+(I-1-4*INT((I-1)/
   4))*2,D)
2030 REM
2040 REM SUBROUTINE D'AFF
   ICHAGE
2050 REM DANS LA POMME
2060 REM
2070 SUBEND
2080 SUB AFC(A,B,I,J,T$()
   )
2090 CALL CHAR(J+2,T$(B))
   :: CALL CHAR(I,"0")
2100 IF A<10 THEN 2110 EL
   SE CALL CHAR(I,T$(INT(
   A/10)))
2110 CALL CHAR(I+2,T$(A-I
   NT(A/10)*10))
2120 REM
2130 REM SUBROUTINE D'AFF
   ICHAGE
2140 REM DES RESULTATS
2150 REM
2160 SUBEND
2170 SUB AFR(LI,S$,STAJ,S
   TAF)
2180 IMAGE # : ### JUSTE#
     ### FAUSSE#
2190 A$,B$="S" :: IF STAJ
   <=1 THEN A$=" "
2200 IF STAF<=1 THEN B$="
   "
2210 DISPLAY AT(LI,1):USI
   NG 2180:S$,STAJ,A$,STA
   F,B$
2220 SUBEND
2230 END
```

Flag

Jean-Pierre Lartillier

Ce programme écrit en Basic TI vous propose de reconnaître le drapeau d'un pays. Pour cela, il affiche, bien sûr, le drapeau en question, mais aussi une carte miniature.

Si vous ne trouvez pas la réponse au premier essai, le programme affiche un des caractères constituant le nom du pays ; il fera la même chose jusqu'à ce que vous trouviez, ou que tous les caractères soient dévoilés.

Une fois que vous aurez trouvé le pays, le programme vous demandera le nom de sa capitale.

De même que pour le nom du pays, le programme vous aidera si vous ne trouvez pas le nom de la capitale.

Programme "Flag"

Basic TI

```
100 REM   ----------------
   -----
110 REM   ORDINATEUR TI 9
   9 4/A
120 REM   ----------------
   -----
130 REM   BASIC SIMPLE
140 REM   ----------------
   -----
150 CALL CLEAR
160 REM   ~~~~~~~~~~~~~~
170 REM   ~     FLAG    ~
180 REM   ~~~~~~~~~~~~~~
190 REM   LARTILLIER J-P
200 CALL SCREEN(4)
210 PRINT TAB(11);"******
   **"
220 PRINT TAB(11);"**FLAG
   **"
230 PRINT TAB(11);"******
   **"
240 PRINT :::::::::
250 FOR C=1 TO 5
260 CALL COLOR(5,5,1)
270 CALL COLOR(6,5,1)
280 FOR DELAI=1 TO 50
290 NEXT DELAI
295 CALL SOUND(-10,-5,0)
300 CALL COLOR(6,16,1)
310 CALL COLOR(5,16,1)
315 CALL SOUND(-10,-7,0)
320 FOR DELAI=1 TO 50
330 NEXT DELAI
340 CALL COLOR(5,7,1)
350 CALL COLOR(6,7,1)
360 CALL SOUND(100,494,0)
370 FOR DELAI=1 TO 50
380 NEXT DELAI
390 NEXT C
400 CALL COLOR(5,2,1)
```

```
410 CALL COLOR(6,2,1)
415 CALL HCHAR(1,1,36,768
   )
416 CALL VCHAR(1,1,64,768
   )
417 CALL HCHAR(1,1,60,768
   )
418 CALL VCHAR(1,1,32,768
   )
419 PRINT "UNE TOUCHE ?"
420 CALL KEY(0,K,S)
430 IF S=0 THEN 420
440 PRINT "VOULEZ-VOUS VO
   IR LES REGLES (O/N) ?"
450 CALL KEY(0,K1,S1)
460 IF S1=0 THEN 450
470 IF K1=79 THEN 480 ELS
   E 490
480 GOSUB 5310
490 XX=10
500 CALL CLEAR
510 CALL SCREEN(8)
520 CH$="FFFFFFFFFFFFFFFF
   "
530 CALL CHAR(113,CH$)
540 CALL CHAR(121,CH$)
550 CALL CHAR(129,CH$)
560 CALL CHAR(136,CH$)
570 CALL CHAR(145,CH$)
580 CALL CHAR(153,CH$)
590 CALL COLOR(11,12,1)
600 CALL COLOR(12,13,1)
610 CALL COLOR(13,5,1)
620 CALL COLOR(14,16,1)
630 CALL COLOR(15,10,1)
640 CALL COLOR(16,2,1)
650 REM  ---ROND---
660 FOR I=1 TO 13
670 READ CH2$
680 CALL CHAR(I+93,CH2$)
690 NEXT I
700 DATA 0000000000000103
   ,030F1F3F8FFFFFFF,FFFF
   FFFFFFFFFFFF,C0F0F8FCF
   1FFFFFF,000
0000000080C0
710 DATA 07070F0F0F0F0F0F
   ,E0E0F0F0F0F0F0F0,0F0F
   0F0F0F0F0707,F0F0F0F0F
   0F0E0E0,030
1000000000000
720 DATA FFFFFF8F3F1F0F03
   ,FFFFFFFEFCF8F080,C080
   800000000000
730 REM    FRANCE
740 CA1=129
750 CA2=136
760 CA3=145
770 A$="FRANCE"
780 C$="PARIS"
790 P1$="000406E7FF7F3F1F
   "
800 P2$="4060F8FFFFFFFEFC
   "
810 P3$="3F3F7F7FFF7F3F0F
   "
```

```
820 P4$="F8F6FEFEFEFF1E08
   "
830 GOSUB 5500
840 GOSUB 4450
850 GOSUB 4760
860 REM   IRLANDE
870 CA1=121
880 CA2=136
890 CA3=113
900 A$="IRLANDE"
910 C$="BELFAST"
920 P1$="000000181E0E7F3F
   "
930 P2$="047C7FFFFF7FF6F0
   "
940 P3$="3F3F7FFFFFFF8F86
   "
950 P4$="F0F0F0F0F0E0E040
   "
960 GOSUB 5500
970 GOSUB 4450
980 GOSUB 4760
990 REM  ITALIE
1000 CA3=145
1010 P1$="0307FFFEFEFF870
   7"
1020 P2$="00800000000000C
   0"
1030 P3$="070301000000000
   0"
1040 P4$="F8FCFC191818101
   0"
1050 GOSUB 5500
1060 A$="ITALIE"
1070 C$="ROME"
1080 GOSUB 4450
1090 GOSUB 4760
1100 REM  GUINEE
1110 CA1=145
1120 CA2=136
1130 CA3=113
1140 P1$="18387F7FFFFFFFF
   F"
1150 P2$="3030F8FEFEFEFFF
   F"
1160 P3$="7F7F38300000000
   0"
1170 P4$="FFFF7F3F3E0E060
   C"
1180 GOSUB 5500
1190 A$="GUINEE"
1200 C$="CONAKRY"
1210 GOSUB 4450
1220 GOSUB 4760
1230 REM  AFGANISTAN
1240 CA1=153
1250 CA2=145
1260 P1$="00000103033F3F3
   F"
1270 P2$="080CFCFEFFFFFFF
   E"
1280 P3$="3F1F1F3F0000000
   0"
1290 P4$="F0F080800000000
   0"
1300 GOSUB 5500
```

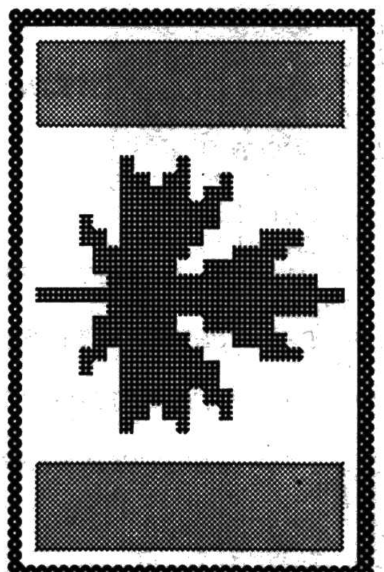
```
1310 CA3=121
1320 A$="AFGANISTAN"
1330 C$="KABOUL"
1340 GOSUB 4450
1350 GOSUB 4760
1360 REM  BELGIQUE
1370 CA2=113
1380 CA3=145
1390 P1$="0000001FFF7F3F1
   F"
1400 P2$="000000C0E0F0F8F
   C"
1410 P3$="1F0F07030100000
   0"
1420 P4$="FCFEFEF8F8F0F03
   8"
1430 GOSUB 5500
1440 A$="BELGIQUE"
1450 C$="BRUXELLES"
1460 GOSUB 4450
1470 GOSUB 4760
1480 REM  COTE D'IVOIRE
1490 CA1=113
1500 CA2=136
1510 CA3=121
1520 P1$="063F3F3F3F7F7F7
   F"
1530 P2$="00C0E0FFFFFEFEF
   E"
1540 P3$="FFFFFFFF3F1F0F0
   C"
1550 P4$="FEFEFEFEFFFF030
   0"
1560 GOSUB 5500
1570 A$="COTE D'IVOIRE"
1580 C$="ABIDJAN"
1590 GOSUB 4450
1600 GOSUB 4760
1610 REM  NIGERIA
1620 CA1=121
1630 CA2=136
1640 CA3=CA1
1650 P1$="007FFFFF7F7F7F7
   F"
1660 P2$="0078FCFFFEFEFCF
   C"
1670 P3$="7F7FFFFFFFF0F0F0
   7"
1680 P4$="FCFCFCFCE4E0C08
   0"
1690 GOSUB 5500
1700 A$="NIGERIA"
1710 C$="LAGOS"
1720 GOSUB 4450
1730 GOSUB 4760
1740 REM  MALI
1750 CA1=121
1760 CA2=113
1770 CA3=145
1780 P1$="030301010101010
   1"
1790 P2$="80E0F0F8FEFFFFF
   F"
1800 P3$="0141FF7FFFC7070
   7"
1810 P4$="FFFFFCF8F0E0E06
```

```
   0"
1820 GOSUB 5500
1830 A$="MALI"
1840 C$="BAMAKO"
1850 GOSUB 4450
1860 GOSUB 4760
1870 REM  ALLEMAGNE
1880 CA1=153
1890 CA2=145
1900 CA3=113
1910 P1$="070F0F1F7F3F3F1
   F"
1920 P2$="80E0F0E0C0C0C0E
   0"
1930 P3$="1F1F3F7FFFFFFF0
   0"
1940 P4$="F0F8FCFFFFFEFE3
   C"
1950 GOSUB 5500
1960 A$="ALLEMAGNE"
1970 C$="BONN"
1980 GOSUB 4520
1990 GOSUB 4760
2000 REM  ARGENTINE
2010 CA1=129
2020 CA2=136
2030 CA3=CA1
2040 P1$="000F0F0F0F0F0F0
   F"
2050 P2$="1830E0C0E0E0000
   0"
2060 P3$="0F0F0F0F0F0F070
   1"
2070 P4$="0000C0800000C0E
   0"
2080 GOSUB 5500
2090 A$="ARGENTINE"
2100 C$="BUENOS AIRES"
2110 GOSUB 4520
2120 GOSUB 4760
2130 REM  AUTRICHE
2140 CA1=145
2150 CA2=136
2160 CA3=CA1
2170 P1$="000000000010F0
   F"
2180 P2$="0000000080F8FFF
   F"
2190 P3$="0F0FFFFF1F07010
   0"
2200 P4$="FEFEFEFEFCC0800
   0"
2210 GOSUB 5500
2220 A$="AUTRICHE"
2230 C$="VIENNE"
2240 GOSUB 4520
2250 GOSUB 4760
2260 REM  GABON
2270 CA1=121
2280 CA2=113
2290 CA3=129
2300 P1$="070707FF7FFFFFF
   F"
2310 P2$="C0E0C0F8F0F0F0F
   8"
2320 P3$="FFFFFF7F3C1C0C0
   8"
2330 P4$="FCFC78080000000
   0"
2340 GOSUB 5500
2350 A$="GABON"
2360 C$="LIBREVILLE"
2370 GOSUB 4520
2380 GOSUB 4760
2390 REM  HAUTE VOLTA
2400 CA1=153
2410 CA2=136
2420 CA3=145
2430 P1$="0000030F1F3F7FF
   F"
2440 P2$="00E0F0F8FCFCFFF
   F"
2450 P3$="7F7FFFFCFC7C040
   4"
2460 P4$="FFFCFC080000000
   0"
2470 GOSUB 5500
2480 A$="HAUTE VOLTA"
2490 C$="OUAGADOUGOU"
2500 GOSUB 4520
2510 GOSUB 4760
2520 REM  HONGRIE
2530 CA1=145
2540 CA2=136
2550 CA3=121
2560 P1$="000000000001C7F
   F"
2570 P2$="0000000078FEFFF
   F"
2580 P3$="FFFFFFFF3F3F1F0
   6"
2590 P4$="FEFCFCFCFCF8800
   0"
2600 GOSUB 5500
2610 A$="HONGRIE"
2620 C$="BUDAPEST"
2630 GOSUB 4520
2640 GOSUB 4760
2650 REM  IRAN
2660 CA1=121
2670 CA2=136
2680 CA3=145
2690 A$="IRAN"
2700 C$="TEHERAN"
2710 P1$="060702027F7F3F3
   F"
2720 P2$="00000000E0F0F0F
   0"
2730 P3$="3F3F1F1F1F0F0E0
   C"
2740 P4$="F8F8FCFCFEFCF80
   0"
2750 GOSUB 5500
2760 GOSUB 4520
2770 GOSUB 4760
2780 REM  LUXEMBOURG
2790 CA1=145
2800 CA2=136
2810 CA3=129
2820 P1$="00000000001030
   3"
2830 P2$="0000000080C0C0E
   0"
2840 P3$="030300000000000
   0"
2850 P4$="C0800000000000
   0"
2860 GOSUB 5500
2870 A$="LUXEMBOURG"
2880 C$=A$
2890 GOSUB 4520
2900 GOSUB 4760
2910 REM  LYBIE
2920 CA1=145
2930 CA2=136
2940 CA3=153
2950 P1$="787C7EFFFF7F7F3
   F"
2960 P2$="1F1F1EFEFEFEFEF
   E"
2970 P3$="3F3F3F7F3F0F070
   2"
2980 P4$="FEFEFFFFFFFFFF0
   E"
2990 GOSUB 5500
3000 A$="LYBIE"
3010 C$="TRIPOLI"
3020 GOSUB 4520
3030 GOSUB 4760
3040 REM  SALVADOR
3050 CA1=129
3060 CA2=136
3070 P1$="00000000003C3F3
   F"
3080 P2$="00000000000080F
   8"
3090 P3$="3F3F1F0F0100000
   0"
3100 P4$="FCFCF8F0E030000
   0"
3110 GOSUB 5500
3120 CA3=CA1
3130 A$="SALVADOR"
3140 C$="SAN "&A$
3150 GOSUB 4520
3160 GOSUB 4760
3170 REM  GUATEMALA
3180 CA1=129
3190 CA2=136
3200 CA3=129
3210 P1$="0000001030F030
   1"
3220 P2$="00000000E0E0E0E
   0"
3230 P3$="0F0F0F0F0F07030
   2"
3240 P4$="F0F8F0E0C080000
   0"
3250 GOSUB 5500
3260 A$="GUATEMALA"
3270 C$=A$
3280 GOSUB 4450
3290 GOSUB 4760
3300 REM  TCHAD
3310 CA1=129
3320 CA2=113
3330 CA3=145
3340 P1$="06070F0F0F1F3F3
```

```
   F"
3350 P2$="0080C0F8FEFFFFF
   F"
3360 P3$="0F1F1F1F3F3F1F0
   F"
3370 P4$="FFFEFCFCFEFFFC8
   0"
3380 GOSUB 5500
3390 GOSUB 4450
3400 A$="TCHAD"
3410 C$="N'DJAMENA"
3420 GOSUB 4760
3430 REM  BANGLA DESH
3440 XX=10
3450 CALL COLOR(8,12,13)
3460 CALL COLOR(9,12,13)
3470 CALL COLOR(10,12,13)
3480 CA1=121
3490 P1$="0000000000101010
   7"
3500 P2$="7EFEFEF8F8FCFCF
   8"
3510 P3$="0707FF7F7FFF010
   1"
3520 P4$="F8F8F0C0C0E0E0E
   0"
3530 GOSUB 5500
3540 GOSUB 4710
3550 GOSUB 5130
3560 A$="BANGLA DESH"
3570 C$="DACCA"
3580 GOSUB 4760
3590 REM  JAPON
3600 CALL COLOR(8,10,16)
3610 CALL COLOR(9,10,16)
3620 CALL COLOR(10,10,16)
3630 CA1=136
3640 P1$="010000000000000
   0"
3650 P2$="C0C000C0C06060E
   0"
3660 P3$="000101030F0C080
   0"
3670 P4$="E0F0E0E08080000
   0"
3680 GOSUB 5500
3690 GOSUB 4710
3700 GOSUB 5130
3710 A$="JAPON"
3720 C$="TOKYO"
3730 GOSUB 4760
3740 REM  NIGER
3750 CALL COLOR(8,12,16)
3760 CALL COLOR(9,12,16)
3770 CALL COLOR(10,12,16)
3780 CA1=113
3790 CA2=136
3800 CA3=121
3810 P1$="010307070F3F3F3
   F"
3820 P2$="C8FCFEFEFEFEFFF
   F"
3830 P3$="3F7FFFFFFF78381
   8"
3840 P4$="FFFEFEFCF8F8000
   0"
3850 GOSUB 5500
3860 GOSUB 4520
3870 GOSUB 5130
3880 A$="NIGER"
3890 C$="NIAMEY"
3900 GOSUB 4760
3910 REM  INDONESIE
3920 CA1=145
3930 CA2=136
3940 P1$="000F0810101F1F3
   E"
3950 P2$="10F0000000C0000
   0"
3960 P3$="3F1B11111010100
   0"
3970 P4$="808080800000000
   0"
3980 GOSUB 5500
3990 GOSUB 4650
4000 A$="INDONESIE"
4010 C$="DJAKARTA"
4020 GOSUB 4760
4030 REM  LIECHTENSTEIN
4040 CA1=129
4050 CA2=145
4060 P1$="000000001C1F1F1
   F"
4070 P2$="00000000008080E
   0"
4080 P3$="070000000000000
   0"
4090 P4$="F8F010000000000
   0"
4100 GOSUB 5500
4110 GOSUB 4650
4120 A$="LIECHTENSTEIN"
4130 C$="VADUZ"
4140 GOSUB 4760
4150 REM  POLOGNE
4160 CA1=136
4170 CA2=145
4180 P1$="0EFFFFFFFFFFFFF
   F"
4190 P2$="0EFEFFFFFEFEFEF
   E"
4200 P3$="FFFF3F1F0B01000
   0"
4210 P4$="FEFFFEFEFEFE7E0
   1"
4220 GOSUB 5500
4230 GOSUB 4650
4240 A$="POLOGNE"
4250 C$="VARSOVIE"
4260 GOSUB 4760
4270 REM
4280 REM  SUISSE
4290 XX=10
4300 CA1=145
4310 P1$="00000000183F7FF
   F"
4320 P2$="0000000000E0E0E
   0"
4330 P3$="7F3F3F333703000
   0"
4340 P4$="FEFEFC9C9E00000
   0"
4350 GOSUB 5500
4360 GOSUB 4710
4370 CALL VCHAR(7,15,136,
   7)
4380 CALL HCHAR(10,12,136
   ,7)
4390 A$="SUISSE"
4400 C$="BERNE"
4410 GOSUB 4760
4420 GOTO 5590
4430 REM  SOUS PROGRAMME
4440 REM  VERTICAL
4450 FOR I=5 TO 9
4460 CALL VCHAR(5,5+I,CA1
   ,XX)
4470 CALL VCHAR(5,10+I,CA
   2,XX)
4480 CALL VCHAR(5,15+I,CA
   3,XX)
4490 NEXT I
4500 RETURN
4510 REM  HORIZONTAL
4520 FOR I=1 TO 3
4530 CALL HCHAR(4+I,12,CA
   1,XX)
4540 CALL HCHAR(7+I,12,CA
   2,XX)
4550 CALL HCHAR(10+I,12,C
   A3,XX)
4560 NEXT I
4570 RETURN
4580 REM  VERTICAL 2 COUL
   EURS
4590 FOR I=1 TO 11
4600 CALL VCHAR(5,5+I,CA1
   ,XX)
4610 CALL VCHAR(5,17+I,CA
   2,XX)
4620 NEXT I
4630 RETURN
4640 REM  HORIZONTAL 2 CO
   ULEURS
4650 FOR I=1 TO 5
4660 CALL HCHAR(4+I,12,CA
   1,XX)
4670 CALL HCHAR(9+I,12,CA
   2,XX)
4680 NEXT I
4690 RETURN
4700 REM  UNIS
4710 FOR I=1 TO 11
4720 CALL HCHAR(4+I,10,CA
   1,XX+1)
4730 NEXT I
4740 RETURN
4750 REM  REPONSE
4760 X$="QUEL EST CE PAYS
   ?"
4770 FOR G1=1 TO 2
4780 SC=5
4790 FOR G2=1 TO 5
4800 FOR J1=1 TO LEN(X$)
4810 CALL HCHAR(17,5+J1,A
   SC(SEG$(X$,J1,1)))
4820 NEXT J1
4830 FOR DELAI=1 TO 300
```

```
4840 NEXT DELAI
4850 Z$="VOTRE REPONSE ?
          "
4860 FOR J2=1 TO LEN(Z$)
4870 CALL HCHAR(17,5+J2,A
   SC(SEG$(Z$,J2,1)))
4880 NEXT J2
4890 INPUT "            ":R
   EP$
4900 CALL HCHAR(16,4,32,2
   8)
4910 IF REP$=A$ THEN 4990
    ELSE 4920
4920 FOR J3=1 TO G2
4930 CALL HCHAR(19,5+J3,A
   SC(SEG$(A$,J3,1)))
4940 NEXT J3
4950 FOR DELAI=1 TO 200
4960 NEXT DELAI
4970 SC=SC-1
4980 NEXT G2
4990 TOTAL=TOTAL+SC
5000 X$="QUELLE EST SA CA
   PITALE ?"
5010 G$="BONNE REPONSE !!
   !   "
5020 FOR J4=1 TO LEN(G$)
5030 CALL HCHAR(17,5+J4,A
   SC(SEG$(G$,J4,1)))
5040 NEXT J4
5050 FOR DELAI=1 TO 500
5060 NEXT DELAI
5070 CALL CLEAR
5080 A$=C$
5090 NEXT G1
5100 CALL CLEAR
5110 RETURN
5120 REM    ROND
5130 CALL HCHAR(8,13,94)
5140 CALL HCHAR(8,14,95)
5150 CALL HCHAR(8,15,96)
5160 CALL HCHAR(8,16,97)
5170 CALL HCHAR(8,17,98)
5180 CALL HCHAR(9,13,99)
5190 CALL HCHAR(9,14,96,3
   )
5200 CALL HCHAR(9,17,100)
5210 CALL HCHAR(10,13,101
   )
5220 CALL HCHAR(10,14,96,
   3)
5230 CALL HCHAR(10,17,102
   )
5240 CALL HCHAR(11,13,103
   )
5250 CALL HCHAR(11,14,104
   )
5260 CALL HCHAR(11,15,96)
5270 CALL HCHAR(11,16,105
   )
5280 CALL HCHAR(11,17,106
   )
5290 RETURN
5300 REM    REGLES DU JEU

5310 FOR I=200 TO 400 STE
   P 20
5320 CALL SOUND(100,I,0)
5330 NEXT I
5340 CALL CLEAR
5350 PRINT "CE JEU TRES S
   IMPLE"::
5360 PRINT "CONSISTE A RE
   CONNAITRE "::
5370 PRINT "LE PAYS PUIS
   D'EN DONNER SA CAPITAL
   E."
5380 PRINT ::
5390 INPUT "enter":XX$
5400 CALL CLEAR
5410 PRINT "VOTRE ORDINAT
   EUR POUR VOUS AIDER"
5420 PRINT "VOUS DONNERA
   LA CARTE DE CE PAYS ET
    SON DRAPEAU"
5430 PRINT :::
5440 PRINT "APPUYER SUR U
   NE TOUCHE !"
5450 CALL KEY(0,K,S)
5460 IF S=0 THEN 5450
5470 CALL CLEAR
5480 RETURN
5490 REM    IMPRESSION CAR
   TES
5500 CALL CHAR(130,P1$)
5510 CALL CHAR(131,P2$)
5520 CALL CHAR(132,P3$)
```



```
5530 CALL CHAR(133,P4$)
5540 CALL HCHAR(2,16,130)
5550 CALL HCHAR(2,17,131)
5560 CALL HCHAR(3,16,132)
5570 CALL HCHAR(3,17,133)
5580 RETURN
5590 REM    RESULTATS
5600 REM  ACCOMPAGNEMENT
5610 FOR I=1 TO 2
5620 CALL SOUND(100,440,5
   ,880,5)
5630 CALL SOUND(100,466,5
   ,930,5)
5640 NEXT I
5650 CALL SOUND(800,440,2
   ,880,2)
5660 CALL SOUND(200,392,0
   ,784,0)
5670 CALL SOUND(200,349,0
   ,698,0)
5680 CALL SOUND(200,330,0
   ,659,0)
5690 CALL SOUND(200,294,0
   ,587,0)
5700 CALL SOUND(800,277,0
   ,554,0)
5710 CALL SOUND(1000,294,
   0,587,0)
5720 CALL CLEAR
5730 PRINT "VOUS AVEZ TOT
   ALISE ";TOTAL;" POINTS
   .."
5740 PRINT ::
5750 PRINT TAB(18);"SUR..
   .260"
5760 IF TOTAL=260 THEN 57
   70 ELSE 5800
5770 PRINT "****EXCELLENT
   S RESULTATS****"::::
5780 INPUT "ENTER POUR RE
   JOUER      FCTN CLEAR
   POUR ARRETER":XX$
5790 GOTO 150
5800 IF TOTAL>3*260/4 THE
   N 5810 ELSE 5840
5810 PRINT "RESULTATS ASS
   EZ CORRECTS..."::::
5820 INPUT "ENTER POUR RE
   JOUER      FTCN CLEAR
   POUR ARRETER":XX$
5830 GOTO 150
5840 IF TOTAL>180 THEN 58
   50 ELSE 5880
5850 PRINT "RESULTATS MOY
   ENS"::::
5860 INPUT "ENTER POUR RE
   JOUER      FCTN CLEAR P
   OUR ARRETER":XX$
5870 GOTO 150
5880 IF TOTAL>65 THEN 589
   0 ELSE 5920
5890 PRINT "VOUS AVEZ DE
   GROS PROGRES A FAIRE":
   :::"ESSAYEZ UNE NOUVEL
   LE FOIS!!!"::
5900 INPUT "ENTER POUR RE
   JOUER      FCTN CLEAR P
   OUR ARRETER":XX$
5910 GOTO 150
5920 PRINT "C'EST INCROYA
   BLE. RESULTATS TRES DE
   CEVANTS!!!"::::
5930 INPUT "ENTER POUR RE
   JOUER      FCTN CLEAR P
   OUR ARRETER":XX$
5940 GOTO 150
5950 END
```

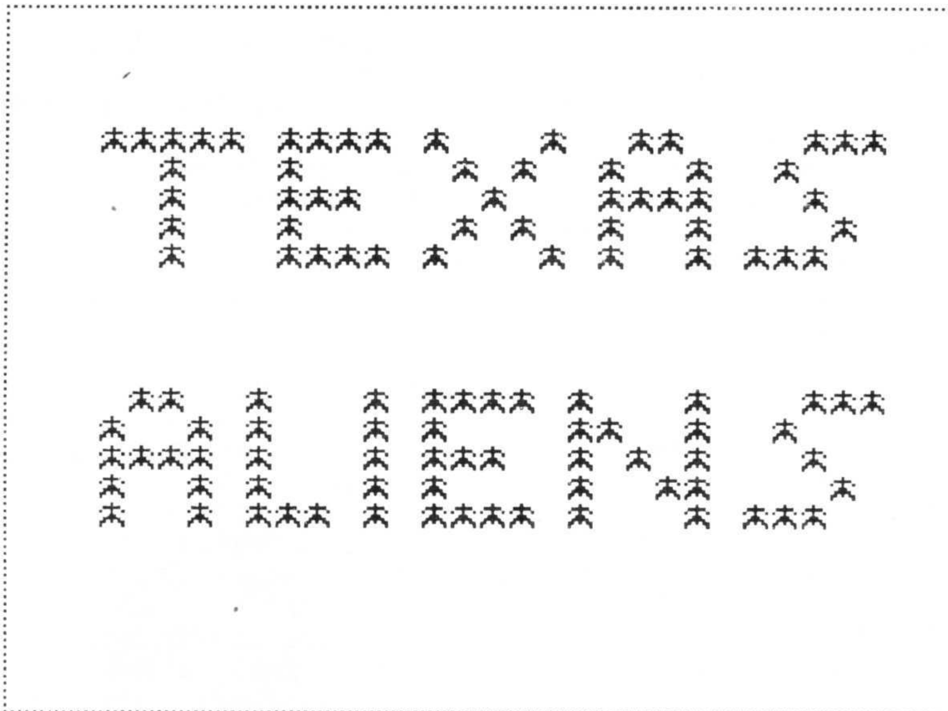
Texas Aliens

Christian Lucas

Le but de ce jeu d'adresse, écrit en Basic Étendu, est simple : détruire le maximum d'"envahisseurs' avant que l'un deux n'atterrisse.

Pour détruire un envahisseur, placez vous à sa verticale en vous aidant des touches fléchées 'S' et 'D', et tirez en appuyant sur la touche '.' ('>'). La destruction d'un envahisseur provoque l'apparition d'un nouvel ennemi en haut de l'écran. Ce dernier descend un peu plus vite que celui que vous venez de détruire.

Le jeu est volontairement lent afin que les parties durent plus de deux minutes.



La programmation

Pour accélérer le tracé du sol et des étoiles, plutôt qu'utiliser la fonction RND, le programme lit des 'DATA' issues de la fonction RND.

Les lignes 860 à 910 évitent les codes hexadécimaux à ralonge.

Les lignes 1270 à 1300 constituent le cœur du programme ; elles doivent être éditées en deux fois avec la méthode habituelle : à la saturation du buffer, faire <ENTER>, puis rééditer la ligne et la compléter.

Le programme conserve le meilleur score pendant toute la durée du jeu.

99

Programme "Texas Aliens"

Basic Étendu

```
100 !********************
110 !*                  *
120 !*  TEXAS ALIENS    *
130 !*                  *
140 !*  SUR TI 99/4A    *
150 !*                  *
160 !* EN BASIC ETENDU  *
170 !*                  *
180 !* CHRISTIAN LUCAS  *
190 !*                  *
200 !*  DEDIE A MAGUY   *
210 !*                  *
220 !********************
230 CALL CLEAR
240 DISPLAY AT(10,3):"Il
   y a longtemps dans une
   "
250 DISPLAY AT(12,7):"gal
   axie lointaine"
260 FOR I=1 TO 400
270 NEXT I
280 CALL CLEAR
290 STAR=0
300 DEF RN=INT(RND*27)+1
310 CALL CHAR(91,"0010204
   0FF402010")! FLECHE S
320 CALL CHAR(93,"0008040
   2FF020408")! FLECHE D
330 CALL CHAR(112,"080808
   0808080808")!LASER
340 CALL CHAR(120,"000808
   040FFF001F1F387870303E
   1E0600101020F0FF00F8F8
   1C1E0E0C7C7860")!#3 EN
   VAHISSEUR JAUNE CLAIR
350 CALL CHAR(124,"010709
   030703010143476F7F6741
   400080E090C0E0C08080C2
   E2F6FEE6820200")!#1 CA
   NON LASER BLANC
360 CALL CHAR(128,"080423
   110F03071F3F3878787060
   60601020C488F0C0E0F8FC
   1C1E1E0E060606")!#5 EN
   VAHISSEUR MAGENTA
370 CALL CHAR(132,"183C3C
   3C3C180F1F3F4008420842
   0840183C3C3C3C18F0F8FC
   02104210421002")!#4 EN
```

```
   VAHISSEUR VERT MOYEN
380 CALL CHAR(136,"000000
   010F7FFFFE3FFFFF7F12
   24C000000080F0FEFFFC7
   FFFFFFFE482403")!#2 EN
   VAHISSEUR ROUGE CLAIR
390 CALL CHAR(140,"8"&RPT
   $("0",63))!#6 A #20 ET
   OILES
400 CALL SCREEN(2):: CALL
    CHAR(64,"107C92387CD6
   8200")
410 PRINT " @@@@@ @@@@ @
     @  @@     @@@"
420 PRINT "   @   @      @
    @  @  @  @"
430 PRINT "   @   @@@
   @  @@@@   @"
440 PRINT "   @   @      @
    @  @  @    @"
450 PRINT "   @   @@@@ @
     @ @  @ @@@"
460 PRINT : : : :
470 PRINT "  @@  @   @ @@
   @@ @   @   @@@"
480 PRINT " @  @ @   @ @
      @@  @  @"
490 PRINT " @@@@ @   @ @@
   @  @ @ @   @"
500 PRINT " @  @ @   @ @
      @  @@    @"
510 PRINT " @  @ @@@ @ @@
   @@ @   @ @@@"
520 PRINT : : : : :
530 CALL COLOR(5,14,1)
540 RESTORE
550 FOR I=1 TO 9
560 READ A,B
570 CALL SOUND(A,B,1)
580 NEXT I
590 FOR I=1 TO 100
600 NEXT I !WAIT
610 FOR I=1 TO 9
620 READ A,B
630 CALL SOUND(A,B,1)
640 NEXT I
650 CALL CLEAR
660 FOR I=2 TO 8
670 CALL COLOR(I,14,1)
680 NEXT I
690 CALL COLOR(9,11,1)
700 CALL COLOR(11,6,1)
710 CALL MAGNIFY(4)
720 CALL SPRITE(#1,132,10
   ,80,85)
730 CALL SPRITE(#2,136,8,
   80,152)
740 DISPLAY AT(5,8):"* TE
   XAS-ALIENS *"
750 DISPLAY AT(18,2):"[ :
   S    ] : D    TIR :
    ."
760 DISPLAY AT(22,6):"(C)
   CHRISTIAN LUCAS"
770 DISPLAY AT(24,5):"APP
   UYER SUR UNE TOUCHE"
780 CALL KEY(0,K,S)
790 IF S=0 THEN 780
800 CALL DELSPRITE(ALL)
810 RANDOMIZE
820 N=0
830 CALL CLEAR
840 CALL SCREEN(2)
850 CALL MAGNIFY(3)
860 ! DESSIN DU SOL
870 A$=RPT$("0",12)
880 CALL CHAR(100,A$&"E6F
    F")
890 CALL CHAR(101,A$&"FFF
    F")
900 CALL CHAR(102,A$&"00F
    F")
910 CALL CHAR(103,A$&"F9F
    F")
920 ! ************
930 RESTORE 1540
940 FOR I=1 TO 32
950 READ X
960 CALL HCHAR(24,I,100+X
    )
970 NEXT I
980 CALL SPRITE(#2,136,10
    ,1,37):: V2=4
990 CALL SPRITE(#3,120,12
    ,1,85):: V3=10
1000 CALL SPRITE(#4,132,3
    ,1,165):: V4=20
1010 CALL SPRITE(#5,128,1
    4,1,213):: V5=26
1020 CALL SPRITE(#1,124,1
    6,176,127):: V=15
1030 IF STAR=1 THEN 1080
    ELSE STAR=1
1040 RESTORE 1560
1050 FOR I=6 TO 28 :: REA
    D A,B,C
1060 CALL SPRITE(#I,140,A
    ,B,C)
1070 NEXT I
1080 RESTORE 1500
1090 FOR I=1 TO 13
1100 READ A,B
1110 CALL SOUND(A,B,1)
1120 NEXT I
1130 FOR I=2 TO 5 :: CALL
     MOTION(#I,1,0):: NEXT
     I :: S2,S3,S4,S5=1
1140 GOTO 1160
1150 N=N+1 :: CALL VCHAR(
    1,V+2,32,22):: GOTO 12
    00
1160 CALL KEY(0,K,S)
1170 IF K=83 THEN 1220 !S
1180 IF K=68 THEN 1240 !D
1190 IF K=46 THEN 1260 !.
1200 CALL POSITION(#2,Y2,
    X,#3,Y3,X,#4,Y4,X,#5,Y
    5,X)
1210 IF Y2>167 OR Y3>168
    OR Y4>169 OR Y5>170 TH
    EN CALL MOTION(#1,0,0,
    #2,0,0,#3,0,0,#4,0,0,#
    5,0,0):: GOTO 1330 ELS
    E 1160
1220 IF V>1 THEN V=V-1 ::
     CALL LOCATE(#1,176,V*
    8+5)
1230 GOTO 1160
1240 IF V<29 THEN V=V+1 :
    : CALL LOCATE(#1,176,V
    *8+5)
1250 GOTO 1160
1260 CALL SOUND(10,800,2)
    :: CALL SOUND(5,400,8)
1270 IF V=V2 THEN CALL PO
    SITION(#2,X,Y):: CALL
    VCHAR(X/8+2,V+2,112,MA
    X(0,20-X/8)):: V2=RN :
    : CALL LOCATE(#2,1,8*V
    2+5):: S2=S2+.1 :: CAL
    L MOTION(#2,S2,0):: GO
    TO 1150
1280 IF V=V3 THEN CALL PO
    SITION(#3,X,Y):: CALL
    VCHAR(X/8+2,V+2,112,MA
    X(0,20-X/8)):: V3=RN :
    : CALL LOCATE(#3,1,8*V
    3+5):: S3=S3+.1 :: CAL
    L MOTION(#3,S3,0):: GO
    TO 1150
1290 IF V=V4 THEN CALL PO
    SITION(#4,X,Y):: CALL
    VCHAR(X/8+2,V+2,112,MA
    X(0,20-X/8)):: V4=RN :
    : CALL LOCATE(#4,1,8*V
    4+5):: S4=S4+.1 :: CAL
    L MOTION(#4,S4,0):: GO
    TO 1150
1300 IF V=V5 THEN CALL PO
    SITION(#5,X,Y):: CALL
    VCHAR(X/8+2,V+2,112,MA
    X(0,20-X/8)):: V5=RN :
    : CALL LOCATE(#5,1,8*V
    5+5):: S5=S5+.1 :: CAL
    L MOTION(#5,S5,0):: GO
    TO 1150
1310 CALL VCHAR(1,V+2,112
    ,22):: CALL VCHAR(1,V+
    2,32,22)
1320 GOTO 1200
1330 RESTORE 1520
1340 FOR I=1 TO 11
1350 READ A,B
1360 CALL SOUND(A,B,1)
1370 NEXT I
1380 CALL DELSPRITE(#1,#2
    ,#3,#4,#5)
1390 DISPLAY AT(7,5):"VOU
    S AVEZ DETRUIT "&STR$(
    N)
1400 DISPLAY AT(9,9):"ENV
    AHISSEURS"
1410 IF N>HI THEN HI=N ::
     DISPLAY AT(12,1):"VOU
    S AVEZ BATTU LE HI-SCO
    RE" ELSE DISPLAY AT(12
    ,5):"LE HI-SCORE EST D
    E "&STR$(HI)
```

```
1420 DISPLAY AT(20,1):"UNE AUTRE PARTIE ( O/N ) ?"
1430 CALL KEY(0,K,S)
1440 IF K=79 THEN 820
1450 IF K=78 THEN CALL CLEAR :: END
1460 GOTO 1430
1470 ! DONNEES MUSICALES
1480 DATA 400,165,400,165,400,165,300,131,100,175,400,165,300,131,100,175,400,165
1490 DATA 400,659,400,659,400,659,300,698,100,659,400,523,300,262,100,349,400,330
1500 DATA 80,175,466,175,466,147,80,175,466,175,466,147
1510 DATA 80,208,466,208,250,196,91,208,91,196,216,175,466,147
1520 DATA 800,147,500,147,200,147,500,147,400,175,200,165,500,165,200,147,500,147,200,139,700,147
1530 REM RANDOM SOL
1540 DATA 3,0,1,1,0,2,2,1,3,1,1,3,1,3,3,0,3,3,2,3,2,0,1,0,1,0,1,0,1,2,0,3
1550 REM RANDOM ETOILE
1560 DATA 14,46,113,7,47,143,11,97,217,7,89,219,7,165,226
1570 DATA 5,168,211,12,142,148,5,95,31,7,52,85
1580 DATA 4,74,176,3,149,29,11,174,37,11,115,222,4,113,199,16,148,132
1590 DATA 16,127,87,7,162,143,16,129,129,16,150,227,5,92,42,4,118,155,5,103,207,16,155,111
```

Un connecteur multiple

Jean Penné

Généralités

Les modules "Solid State Software" de Texas Instruments peuvent contenir deux sortes de composants de mémoire : tout module contient au moins un boîtier GROM de 6Ko (et au plus 5, soit 30Ko). Bien des modules contiennent en plus des boîtiers ROM de type classique (8Ko, ou plus, avec, dans ce dernier cas, un système de sélection des pages).
L'adressse de base de cette page de ROM est >6000.

Seules les ROMs sont programmées en langage machine car la logique d'accès aux GROMs **ne permet pas** au microprocesseur TMS 9900 d'exécuter directement des instructions qui auraient été enregistrées en GROM. Les GROMs contiennent donc seulement deux sortes d'informations :

- des instructions en code "GPL" qui sont interprétées par le programme moniteur de la console, lequel joue le jeu de l'accès au GROMs,
- des données (chaînes de caractères, tables, adresses...)

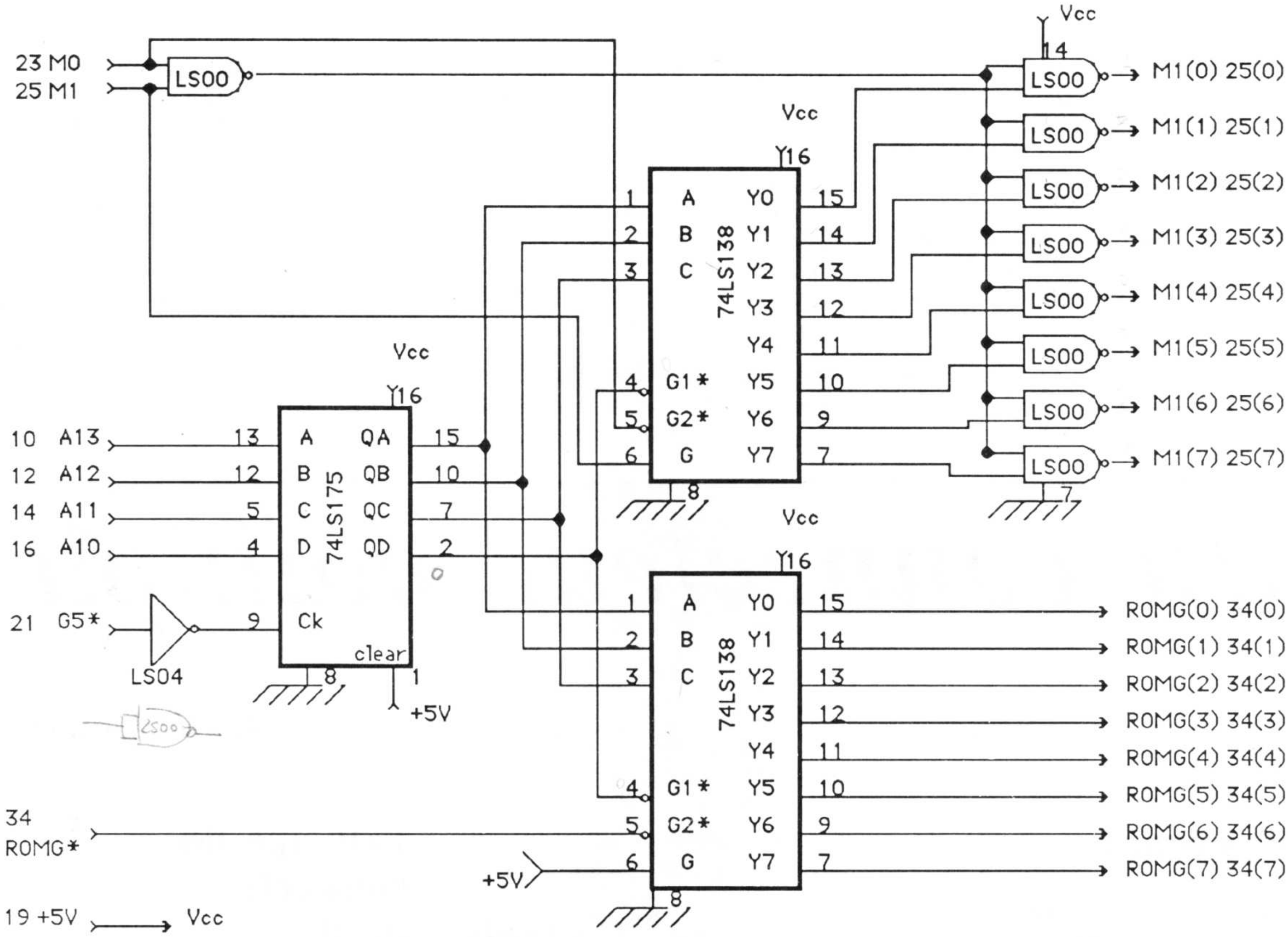
Accès aux GROMs

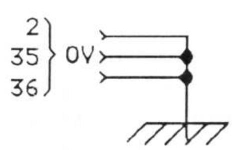
Il se fait par un petit nombre d'adresses fixes (ports) :
GRMRD = >9800 pour lire un octet de donnée,
GRMWA = >9C02 pour écrire la "GROM adress" (en deux fois car ce sont des ports de 8 bits...).
Ces adresses ne sont pas décodées complètement par le matériel de la console, par exemple : >9804 correspond encore à GRMRD. Cependant, le système d'exploitation (moniteur) a été écrit en prévision d'un décodage plus complet, qui permettrait d'avoir accès à 1 parmi 16 modules connectés simultanément sur le bus auxiliaire (GROM port connector).
C'est la raison d'être de la fonction "Review Modulo Library" qui est rendue accessible automatiquement dès que le système a détecté une différence entre deux lecture GROM à la même "GROM adress" mais l'une effectuée par le port >9800 et l'autre par le port >9804.

Principe du connecteur multiple

La réalisation décrite ci-dessous permet d'utiliser cette ressource, jusqu'ici *dormante*, du système pour accéder automatiquement à plusieurs modules branchés simultanément sur le connecteur "bus auxiliaire" de la console.

Les modules peuvent contenir GROM, ROM et même RAM (mini-mémoire par exemple).
Pour cela (voir schéma), les signaux d'adresse A_{10}, A_{11}, A_{12}, A_{13} sont mémorisés dans un circuit 74LS175, au moment de la descente du signal de sélection globale des GROMs : GS*. Pour sélectionner les ROMs, il suffit alors de démultiplexer le signal ROMG* avec ces signaux d'adresses mémorisé, au moyen d'un décodeur 74LS138 (ici limité à 8 module et en fait 6 pour des raisons d'encombrement physique...). En ce qui concerne les GROMs, le problème est un peu plus délicat : un simple démultiplexage de GS* **ne fonctionne pas** ; en effet,





toutes les GROMs présentes dans le système doivent être sélectées ensemble à chaque accès, qu'elles soient lues ou non... Ceci est indispensable pour assurer la mise à jour de leur registre d'adresse interne.

Nous avons donc choisi d'agir sur le signal de commande M1 de sorte que seul le boîtier concerné voie une commande de lecture, tous les autres voyant alors une commande d'écriture, ce qui n'a pas d'inconvénient pour une GROM (mais en aurait pour une GRAM, telle que celle du module Maximem.

Réalisation

Sur une plaque imprimée genre *Verroboard* au pas de 2,54 mm, de dimensions 200 x 100 mm, et comportant 37 bandes de 76 trous, il faut disposer en bout un connecteur mâle 2 x 19 contacts, et, sur la carte, 6 connecteur femelle 2 x 19 contacts (faute de 2 x 18...). Le premier travail consiste à transmettre tels-quels tous les signaux du bus auxiliaire, à l'exception de deux d'entre-eux qui sont : le n° 25 (M1) et le n° 34 (ROMG*), le n° 1 étant en-dessous et à gauche suivant la conventions du *Servicing Manual* de TI.

Il ne reste plus qu'à compléter la carte avec le logique du schéma 1. La place ne manque pas pour y loger les 6 ou 7 circuits intégrés nécessaires.

Fonctionnement

Supposons que vous branchiez en position 0 le module TI LOGOII et en position 1 le module Editor-Assembler. A la mise sous tension, après avoir appuyé sur une touche, vous aurez l'écran 0 :

```
PRESS:
    1 FOR TI BASIC
    2 FOR TI LOGOII
    3 FOR REVIEW MODULE LIBRARY
```

Vous pouvez donc choisir immédiatement TI BASIC ou LOGO.

Si vous appuyer sur la touche 3, vous obtenez alors l'écran :

```
PRESS:
    1 FOR TI BASIC
    2 FOR EDITOR ASSEMBLER
    3 FOR REVIEW MODULE LIBRARY
```

Si vous choisissez l'option 2, vous pourrez bien sûr éditer et assembler mais de plus vous pourrez charger DEBUG et *dumper* la ROM ou la GROM de TI LOGOII ou charger DESA pour la désassembler... Votre système a acquis une dimension de plus.

Avec l'option 3, l'exploration des *banques de modules* continue et comme vous n'avez pas branché d'autre module, vous revenez à l'écran 0.

99

Copie d'écran pour le Basic Étendu

Michel Teyssou

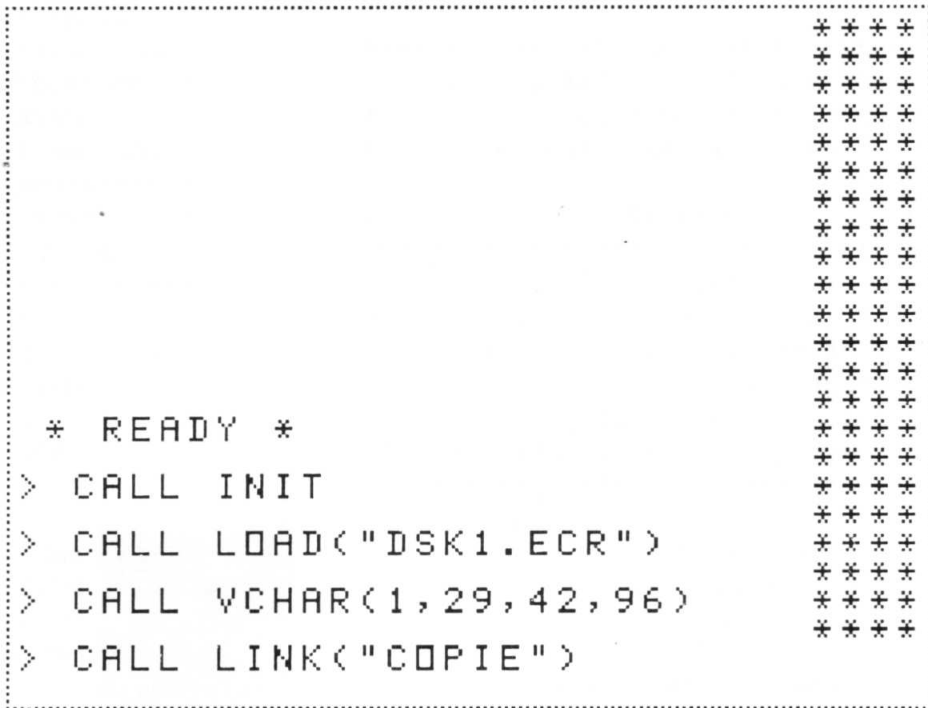
Voici une routine, écrite en assembleur, permettant une copie d'écran graphique à partir du Basic Étendu. Elle fonctionne sur les imprimantes *Epson*, *Mannesman* et *Star*. De plus, elle comporte deux particularités intéressantes :

- un fichier d'inversion d'octets ;
- un fichier DSRLNK.

Il est possible, par exemple, d'incorporer cette routine de copie d'écran au programme de J.F. Guillemant (Comptes familiaux) : il suffit pour cela de mettre CALL LINK("COPIE") à la ligne 2800 ("COPIE" est le nom du programme qui se trouve en DEFinition). Il faudra auparavant qu'elle soit chargée dans l'extension mémoire. Vous mettez par exemple dans un programme baptisé "LOAD" :

```
10 CALL INIT
20 CALL LOAD("DSK1.ECR")
30 RUN "DSK1.COMPTES"
```

"ECR" est le nom que vous aurez donné au fichier objet sur la disquette, et "COMPTES" celui que vous aurez donné au programme écrit en Basic Étendu.



Configuration

Si vous devez adapter la routine en fonction des caractéristiques de votre imprimante, les nombreuses remarques placées dans le fichier source pourront vous guider. Il faut, dans l'ordre, placer :

- les codes (hexadécimaux) pour l'impression unidirectionnelle. Ils sont, selon l'imprimante, sur deux ou trois octets ; compléter éventuellement avec des >00 ;
- les codes correspondants à l'impression de lignes graphiques jointives ; compléter éventuellement avec des >00 ;
- les codes permettant d'obtenir une impression en simple densité (toujours sur deux octets) ;
- immédiatement après ce qui précède, le nombre total d'octets par ligne d'impression (ici 192). Ce nombre est sur deux ou trois octets selon l'imprimante ; compléter éventuellement avec des >00.

Note : si votre imprimante ne reconnaît pas les codes pour l'impression unidirectionnelle (parce que, par exemple, elle ne fonctionne que dans ce mode), mettez des >00 à la place ; ils seront ignorés. Si vous voulez supprimer les octets dont vous ne vous servez pas, il vous faudra modifier le nombre d'octets pris en compte au moment où l'on utilise les étiquettes UNIDIR et GRAPH.

Inversion

Sur les imprimantes citées plus haut, l'octet graphique est, comme sur toute imprimante à impacts, imprimé verticalement, mais avec le bit de poids fort en bas. Avec les imprimantes de type *Seïkosha* ou *ImageWriter d'Apple*, le bit de poids fort est en haut si bien que, sans une routine d'inversion, chaque 'ligne d'impression' serait inversée.

Comme le programme comporte deux possibilités, vous pourrez toujours essayer !

Si votre imprimante imprime 'de haut en bas', il faut supprimer la routine d'inversion. Pour cela, il suffit de mettre des astérisques

(*) devant et d'enlever ceux de la routine normale, comme indiqué dans le source.

DSRLNK

Le fichier DSRLNK est indispensable, car on ne dispose pas, en Basic Étendu, de cette routine située dans le module "Éditeur/assembleur". Pas plus, du reste, que de GPLLNK, qui aurait permis une image mémoire (inversion) plus facile. La routine en question a donc été désassemblée et isolée, puis rendue relogeable. Vous pouvez donc vous en servir dans d'autres réalisations si le besoin s'en fait sentir.

99

Source de la routine de copie d'écran

```
***********************************
* COPIE ECRAN AVEC BASIC ETENDU   *
* 1986  Michel TEYSSOU  Arbus     *
*     64230-LESCAR  (France)      *
*                                 *
*       ET 99 MAGAZINE            *
***********************************
* pour adapter votre imprimante:  *
* il faut modifier les codes du   *
* fichier Imprimante . Mettre     *
* image miroir ou  image normale  *
* selon le cas. (voir plus bas).  *
* Reassembler avec l'option R     *
***********************************
       DEF COPIE
       AORG >2700
* zones tampons
REG    BSS 32
TAMPON BSS 192
* equivalence basic etendu
VSBR   EQU >2028
VSBW   EQU >2020
VMBR   EQU >202C
VMBW   EQU >2024
* concerne le PAB
PAB    EQU >F80
TPAB   EQU >1000
PNTR   EQU >8356
* etiquettes
D192   DATA 192
D8     DATA 8
SAVGPL DATA 0
CAR    DATA 0
TEST   DATA 0
* designation de l'imprimante
PIO    DATA >0012,>1000,>FF00
       DATA >0000,>0004
       TEXT 'PIO.'
**********************************
*  table pour routine inversion *
**********************************
TABLE  BYTE 0,8,4,>C,2,>A,6,>E  *
       BYTE 1,9,5,>D,3,>B,7,>F  *
**********************************
* FICHIER IMPRIMANTE POUR EPSON *
*      OU  MANNESMANN TALLY      *
**********************************
* impression unidirectionelle    *
UNIDIR DATA >1B55,>0100          *
* lignes jointives               *
       DATA >1B41,>0800          *
* mode grahique densite normale  *
GRAPH  DATA >1B4B                *
* nombre d'octet par ligne       *
       DATA >C000,>0000          *
* ici sur 2 octets N1,N2 (192)   *
* certaines imprimantes sont sur*
* 3 octets >31,>39,>32  (192)    *
* "IMAGEWRITER" & "SEIKOSHA"     *
**********************************
       EVEN
DSR    LI R6,PAB+9
       MOV R6,@PNTR
       BLWP @DSRLNK
       DATA 8
       RT

COPIE  LWPI REG
       MOVB @>9802,@SAVGPL
       MOVB @>9802,@SAVGPL+1
       DEC @SAVGPL
* ouvre
       LI R0,PAB
       LI R1,PIO
       LI R2,14
       BLWP @VMBW
       BL @DSR
* print
       LI R0,PAB
       LI R1,>0300
       BLWP @VSBW
       LI R0,TPAB
* unidirection+lignes jointes
       LI R1,UNIDIR
       LI R2,8        Nb octet |dans
       BLWP @VMBW              |l'eti-
       LI R0,PAB+5             |quette
       LI R1,>0800    Nb octet |UNIDIR
       BLWP @VSBW
       BL @DSR
* saisie ecran du code ascii
       LI R4,31
AUTRE  MOV R4,R0
       A @CAR,R0
       CLR R1
       BLWP @VSBR
       SWPB R1
       AI R1,-128
* on x le code par 8
       SLA R1,3
```

```
* aller dans table descripteurs
       LI R0,>400
       A R1,R0
**************************
* Routine avec inversion *
*     (Image miroir)     *
*  EPSON-MANNESMAN TALLY *
**************************
* transfert des 8 octets *
       LI R2,8           *
       LI R3,TAMPON      *
       A @TEST,R3        *
AUT2   CLR R1            *
       BLWP @VSBR        *
       SRL R1,8          *
       MOV R1,R10        *
       SRL R10,4         *
       LI R7,TABLE       *
       A R10,R7          *
       CLR R10           *
       MOVB *R7,R10      *
       ANDI R1,>000F     *
       LI R7,TABLE       *
       A R1,R7           *
       CLR R1            *
       MOVB *R7,R1       *
       SLA R1,4          *
       A R1,R10          *
       MOVB R10,*R3+     *
       INC R0            *
       DEC R2            *
       JNE AUT2          *
**************************Masquer la
* Routine sans inversion *routine qui
*     (Image normale)    *est au dessus
* Remplace la precedente *en mettant des
* (IMAGEWRITER)(SEIKOSHA)*asterisques
**************************devant toutes
* transfert des 8 octets *les lignes et
*      LI R2,8           *mettre celle-
*      LI R1,TAMPON      *ci en service
*      A @TEST,R3        *en enlevant
*      BLWP @VMBR        *les asterisques
**************************des lignes 140
       A @D8,@TEST       *a 143.
       C @TEST,@D192     *Reassembler avec
       JEQ IMPR          *l'option R.
       LI R2,32
       A R2,@CAR
       JMP AUTRE
* impression
* mode graph
IMPR   LI R0,TPAB
       LI R1,GRAPH
       LI R2,6      Nb octet dans GRAPH
       BLWP @VMBW
       LI R0,TPAB+6 TPAB+Nb octet
       LI R1,TAMPON
       LI R2,192
       BLWP @VMBW
       LI R0,PAB+5
```

```
       LI R1,>C600   >C0 + Nb octet(ici 6)
       BLWP @VSBW
       BL @DSR
       CLR @CAR
       CLR @TEST
       DEC R4
       CI R4,-1
       JNE AUTRE
* retour basic
       LI R1,>0100
       LI R0,PAB
       BLWP @VSBW
       BL @DSR
       MOVB @SAVGPL,@>9C02
       MOVB @SAVGPL+1,@>9C02
       CLR @>837C
       LWPI >83E0
       B @>0070
****************************
* FICHIER ADRESSAGE DSRLNK *
****************************
REG1   BSS  38
REG2   BSS  32
DAT1   BSS  6
DAT2   DATA >2000
DAT3   DATA >2EAA
SAV11  DATA >0000
*******************************
VSBR1  DATA REG1                *
       DATA $+2                 *
       CLR  R1                  *
       MOV  *R13,R2             *
       MOVB @REG1+5,@>8C02      *
       SOC  R1,R2               *
       MOVB R2,@>8C02           *
       MOV  @>0002(R13),R1      *
       MOV  @>0004(R13),R2      *
       MOVB @>8800,@>0002(R13)  *
       RTWP                     *
*******************************
DSRLNK DATA REG1+6
       DATA $+2
       MOV  *R14+,R5
       SZCB @DAT2,R15
       MOV  @>8356,R0
       MOV  R0,R9
       AI   R9,>FFF8
       BLWP @VSBR1
       MOVB R1,R3
       SRL  R3,8
       SETO R4
       LI   R2,DAT1
S006   INC  R0
       INC  R4
       C    R4,R3
       JEQ  S005
       BLWP @VSBR1
       MOVB R1,*R2+
       CB   R1,@DAT3
       JNE  S006
S005   MOV  R4,R4
```

```
       JEQ  S007
       CI   R4,>0007
       JGT  S007
       CLR  @>83D0
       MOV  R4,@>8354
       INC  R4
       A    R4,@>8356
       LWPI >83E0
       CLR  R1
       LI   R12,>0F00
S010   MOV  R12,R12
       JEQ  S008
       SBZ  0
S008   AI   R12,>0100
       CLR  @>83D0
       CI   R12,>2000
       JEQ  S009
       MOV  R12,@>83D0
       SBO  0
       LI   R2,>4000
       CB   *R2,@DAT3+1
       JNE  S010
       A    @REG1+16,R2
       JMP  S011
S013   MOV  @>83D2,R2
       SBO  0
S011   MOV  *R2,R2
       JEQ  S010
       MOV  R2,@>83D2
       INCT R2
       MOV  *R2+,R9
       MOVB @>8355,R5
       JEQ  S012
       CB   R5,*R2+
       JNE  S013
       SRL  R5,8
       LI   R6,DAT1
S014   CB   *R6+,*R2+
       JNE  S013
       DEC  R5
       JNE  S014
S012   INC  R1
       BL   *R9
       JMP  S013
       SBZ  0
       LWPI REG1+6
       MOV  R9,R0
       BLWP @VSBR1
       SRL  R1,13
       JNE  S015
       RTWP
S009   LWPI REG1+6
S007   CLR  R1
S015   SWPB R1
       MOVB R1,*R13
       SOCB @DAT2,R15
       RTWP
       MOV  R11,@SAV11
       LWPI REG2
       BLWP @VSBR1
       LWPI >83E0
       JEQ  S016
       MOV  @SAV11,R11
       B    *R11
S016   MOVB @REG2,@>8322
       B    @>00CE
       END
```

Indentation automatique

Gérard Santraille

Quoi de plus pénible, lorsqu'on se replonge dans un programme que l'on a écrit plusieurs mois auparavant, que de ne plus rien comprendre à ce que l'on a fait.

Une qualité fondamentale de tout programme susceptible d'être réutilisé par quelqu'un d'autre (ou par soi-même) est la lisibilité.

Un programme dont les entités logiques sont clairement indentées est d'une lecture beaucoup plus aisée qu'un amas incohérent et brouillon d'instructions et/ou de déclarations.

Néanmoins, il est parfois contraignant de s'efforcer à respecter une marge ou un décalage bien particulier pour chaque type de déclarations. Une telle discipline, souvent trop rigide et trop systématique peut-être nuisible.

Pourquoi ne pas confier cette tâche à un ordinateur qui ne sera aucunement effrayé par l'aspect automatique et répétitif de la tâche à accomplir. Le programme que je vous propose va se charger de cette opération pour vous. Il va lire un fichier source et générer à partir de celui-ci un fichier fortement structuré, aéré et indenté, dont chaque entité logique sera bien mise en évidence...
Lors de la création de vos programmes, vous n'aurez plus à vous soucier de leur présentation et vous pourrez vous concentrer pleinement sur l'algorithme.

L'éditeur de textes du système UCSD implanté sur le TI ne pouvant appréhender que des fichiers d'approximativement 12Ko, il a été nécessaire de découper le programme en trois fichiers différents (INDENT1 à INDENT3). Le fichier-source à compiler (INDENT) ne comporte que les directives d'inclusion : {*I nom_de_fichier} de ces trois fichiers.

Son utilisation est extrêmement simple : donnez le nom du fichier à traiter, le nom du fichier à créer et le programme s'occupe de tout le reste. Notez que le programme rajoute l'extension .TEXT par défaut, il est donc inutile de l'écrire. Si le fichier à traiter ne comporte pas d'extension, rajoutez un point à la fin de son nom. Lors de la saisie du nom du fichier à créer, $ (comme dans la syntaxe du système d'exploitation) correspond au nom du fichier source à traiter. Ainsi, si vous désirez conserver le même nom suivi de .IND tapez juste $.IND. Si vous donnez juste $ comme nom du fichier à créer, le programme rajoutera automatiquement un '2' afin d'éviter que fichiers source et destinataire aient le même nom.

Le traitement d'un fichier étant assez long, le programme affiche le numéro de la ligne en cours d'analyse.

Commentons certains aspects du programme :

- La partie déclarative du programme principal fait largement appel à différents types accessibles par PASCAL. Ainsi, MOTS_CLEF et CONTEXTE sont des ensembles définis explicitement et exhaustivement. Ces ensembles sont ordonnés et il est possible de passer séquentiellement d'un élément à un autre (dans n'importe quel sens) au moyen des fonctions PRED (pour prédécesseur) et SUCC (pour successeur). Notons également la déclaration directe du type de la variable PILE en zone de déclarations des variables. Nous aurions pu définir un type STACK (dans la zone type), puis définir PILE comme étant de type STACK. Notons enfin les tableaux CARSPE et ORTHOGRAPHE dont les indices ne sont pas entiers mais sont les éléments des ensembles CONTEXTE et MOTS_CLEF.

- La fonction booléenne FICHIERS_OUVERTS utilise les directives {*I-} et {$I+} pour éviter que le programme ne provoque une erreur au cas où l'un des fichiers que l'on tente d'ouvrir ou de fermer n'existerait pas. On peut tester le déroulement correct d'une procédure RESET ou REWRITE à l'aide de la variable IORESULT qui vaut zéro si tout s'est bien passé. Comme indiqué en commentaires dans le programme, notez l'emploi de cette fonction. Au lieu d'en faire une procédure (OUVRE_FICHIERS), puis de tester si tout s'est bien déroulé (par la valeur d'une variable booléenne interne à la procédure), il est plus élégant d'en faire directement une fonction booléenne. Pour savoir quelle est la valeur de cette fonction (TRUE ou FALSE), le programme est obligé de l'exécuter entièrement.

- Le principe de l'algorithme du programme en lui-même est assez simple. Il lit caractère par caractère le fichier source, reconnaît les mots réservés et construit (mot par mot) un nouveau fichier dont l'indentation dépend entièrement d'un ensemble de variables booléennes et entières caractérisant le contexte du mot déchiffré (partie déclarative, test, commentaire). Ces variables dépendent elles-même du mot réservé reconnu. Ainsi, selon que le programme rencontre un UNTIL ou un CASE, il n'agira pas de la même façon.

- On peut traiter n'importe quel source PASCAL avec ce programme. Tous les mots sont convertis en majuscules avant d'être déchiffrés : le programme accepte donc tout panachage de minuscules et de masjuscules.

- Un programme découpé en plusieurs fichiers (comme celui-ci) doit être traité fichier par fichier.

Lorsque vous entrerez ce programme, abstenez vous de saisir les commentaires, n'oubliez pas de le découper en trois fichiers et ne vous fatiguez pas à respecter l'identation : il le fera pour vous une fois compilé.

99

Fichier INDENT.TEXT

```
{Fichier INDENT.TEXT}

{Ce fichier contient les directives
 d'insertion du source qui a ete decoupe
 en trois modules INDENT1, INDENT2 et INDENT3}

          {$I INDENT1.TEXT}
          {$I INDENT2.TEXT}
          {$I INDENT3.TEXT}

 {C'est ce fichier qu'il faut compiler}
```

Fichier INDENT1.TEXT

```
{Fichier INDENT1.TEXT}

{(C)opyright 99Magazine et l'auteur ..}

program IDENTATION;

const
      MARGEMAX=50;
      NBCOLMAX=80;
      LONGPILE=100;
      LONGIDEN=8;

type
      MOTS_CLEF=(TERME,BEGIN2,CASE2,CONST2,DO2,ELSE2,
          END2,FOR2,FUNCT2,IF2,LABEL2,OF2,PROC2,
          PROGRAM2,RECORD2,REPEAT2,THEN2,TYPE2,UNTIL2,
          VAR2,WHILE2,WITH2);
      {Tous les mots clef se terminent par
       2 pour que le compilateur ne les
       confondent pas avec les vrais ...
       De plus FUNCTION est devenu FUNCT tout
       comme PROCEDURE est devenu PROC.
       En effet, les identificateurs ne sont reconnus
       que sur 8 caracteres.
       FUNCTION2 serait donc equivalent a FUNCTION}
      CONTEXTE=(PROGRAMME,PARENTHESEE,CHAINE);
      IDENTIFICATEUR=packed array [1..LONGIDEN] of char;

var

      SOMMET:0..LONGPILE;
      TRAITEMENT:MOTS_CLEF;
      PILE:array [1..LONGPILE] of record
                                    RETOUR:integer;
                                    TABULATION:boolean
                                  end;
      CARSPE:array [CONTEXTE] of set of char;
      MOT_A_RECHERCHER:IDENTIFICATEUR;
      TAMPON:string[132];
      ENTREE,SORTIE :interactive;
      ORTHOGRAPHE:array [MOTS_CLEF] of IDENTIFICATEUR;
      SITUATION:CONTEXTE;
      LIGNE,MARGE,NIVEAU,NBCOL,RETARD,NBRPARENTHESES:integer;
      SOUS_DECLAR,DEBUT_PRO,SOUS_PRO,SOUS_CASE,SOUS_END,SOUS_PARENTHESES:
          boolean;
      CHANGER_LIGNE,PILE_VIDE,CAR_SPECIAL:boolean;
      DERNIER_CAR,CAR_SUIVANT:char;

    procedure aff_traitement;

    begin
      ligne:=ligne+1:
      gotoxy(0,20);
      writeln('Ecriture de la ligne',ligne:4,' en cours ..')
    end;

    procedure ERREUR (code: integer);

    begin { erreur }
      case code of
        1:
            write('Erreur de syntaxe');
        2:
```

```
        write('Debordement de pile');
      3:
        write('Fichier vide');
    end;
    write(', Pressez <ENTER>',chr(7));
    readln
  end;

  function FICHIERS_OUVERTS:boolean;

  var
        p:integer;
        abandon:boolean;
        source,destinataire:string;

  begin
    page(output);
    writeln('Identation de sources PASCAL');
    writeln('----------------------------');
    writeln;
    repeat
      writeln('Fichier a traiter',chr(7));
      write   ('.TEXT  par defaut : ');
      readln(source);
      abandon:=source='';
      {abandon sera VRAI si source est vide ..}
      if pos('.',source)=0
        then
          source:=concat(source,'.text');
          {rajoute automatiquement .TEXT
           si aucune extension.
           Entrez un point apres le nom
           pour traiter un fichier sans extension
           Exemple : TOTO.  pour TOTO              }
      if not abandon
        then
        {l'emploi des directives I+ et I-
         permet de ne pas sortir du programme
         si on ne specifie pas un nom de fichier
         valide
         Rappel : ioresult = 0
         si l'operation d'entree-sortie s'est
         convenablement effectuee..}
          {$I-}
          reset(entree,source)
          {$I+}
    until abandon or (ioresult=0);
    if not abandon
      then
        repeat
          writeln('Fichier destinataire',chr(7));
          writeln('.TEXT par defaut');
          write  ('$ = fichier source : ');
          {$ est un joker qui equivaut au nom
           du fichier source. Ainsi $.ID
           donnera TOTO.TEXT.ID si TOTO.TEXT
           etait le source ...}
          readln(destinataire);
          abandon:=destinataire='';
          if destinataire='$'
            then
              destinataire:='$2';
              {on rajoute un 2 pour eviter que
               les fichiers source et destinataire
               aient le meme nom ...}
          p:=pos('$',destinataire);
          if p<>0
            then
              begin
                delete(destinataire,p,1);
                insert(source,destinataire,p)
              end;
          if pos('.',destinataire)=0
            then
              destinataire:=concat(destinataire,'.text');
          if not abandon
            then
              {$I-}
              rewrite(sortie,destinataire);
              {$I+}
        until abandon or (ioresult=0);
    FICHIERS_OUVERTS:=not abandon;
    writeln;
    writeln
  end;

  procedure INITIALISER;

  begin { INITIALISER }
    MARGE :=0;
    NBCOL :=0;
    RETARD:=0;
    NIVEAU:=0;
    SOMMET:=0;
```

```
    LIGNE :=1;

    CHANGER_LIGNE    :=false;
    PILE_VIDE        :=true;
    SOUS_DECLAR      :=false;
    DEBUT_PRO        :=false;
    SOUS_PRO         :=false;
    SOUS_CASE        :=false;
    SOUS_END         :=false;
    SOUS_PARENTHESES:=false;

    SITUATION:=PROGRAMME;

    ORTHOGRAPHE[TERME]      :='         ';
    ORTHOGRAPHE[BEGIN2]     :='BEGIN    ';
    ORTHOGRAPHE[CASE2]      :='CASE     ';
    ORTHOGRAPHE[CONST2]     :='CONST    ';
    ORTHOGRAPHE[DO2]        :='DO       ';
    ORTHOGRAPHE[ELSE2]      :='ELSE     ';
    ORTHOGRAPHE[END2]       :='END      ';
    ORTHOGRAPHE[FOR2]       :='FOR      ';
    ORTHOGRAPHE[FUNCT2]     :='FUNCTION';
    ORTHOGRAPHE[IF2]        :='IF       ';
    ORTHOGRAPHE[LABEL2]     :='LABEL    ';
    ORTHOGRAPHE[OF2]        :='OF       ';
    ORTHOGRAPHE[PROC2]      :='PROCEDUR';
    ORTHOGRAPHE[PROGRAM2]   :='PROGRAM  ';
    ORTHOGRAPHE[RECORD2]    :='RECORD   ';
    ORTHOGRAPHE[REPEAT2]    :='REPEAT   ';
    ORTHOGRAPHE[THEN2]      :='THEN     ';
    ORTHOGRAPHE[TYPE2]      :='TYPE     ';
    ORTHOGRAPHE[UNTIL2]     :='UNTIL    ';
    ORTHOGRAPHE[VAR2]       :='VAR      ';
    ORTHOGRAPHE[WHILE2]     :='WHILE    ';
    ORTHOGRAPHE[WITH2]      :='WITH     ';
    CARSPE[PROGRAMME]:=[' ',';','(','=','{',':',',','.',
        ''''];
    CARSPE[PARENTHESEE]:=[',',';','+','-','/','*','(',')',
        '<','>',''''];
    CARSPE[CHAINE]:=[''''];
    if not eof(ENTREE)
      then
        read(ENTREE,CAR_SUIVANT)
      else
        ERREUR(3)
  end { INITIALISER };

  procedure MEMORISER (CAR : char);

  begin
    tampon:=concat(tampon,' ');
    tampon[length(tampon)]:=car
    {Il n'est pas possible d'ecrire directement
     TAMPON:=concat(TAMPON,car)}
  end;

  procedure TRANSFERT (var MOT_A_RECHERCHER:IDENTIFICATEUR);

  var
        I:integer;

  begin
    if length(TAMPON)>LONGIDEN
      then
        for I:=1 to LONGIDEN do
          MOT_A_RECHERCHER[I]:=TAMPON[I]
      else
        begin
          for i:=1 to length(tampon)do
            mot_a_rechercher[i]:=tampon[i];
          for I:=length(TAMPON)+1 to LONGIDEN do
            MOT_A_RECHERCHER[I]:=' '
        end
  end;

  function CODE_RESERVE (MOT:IDENTIFICATEUR):MOTS_CLEF;

  var
        TRAITEMENT:MOTS_CLEF;

      procedure CONVMAJ (var MOT : IDENTIFICATEUR);

      const
            DECAL=32;
            AMIN =97;
            ZMIN =122;

      var
            I:integer;

      begin { convmaj }
        for I:=1 to LONGIDEN do
          if MOT[I] in [chr(AMIN)..chr(ZMIN)]
            then
```

```
            MOT[I]:=chr(ord(MOT[i])-DECAL)
    END;
begin
  CONVMAJ(MOT);
  TRAITEMENT:=WITH2;
  {Traitement vaut ici WITH2 qui est le dernier
   index du tableau ORTHOGRAPHE. On va remonter
   ce tableau par :
      TRAITEMENT:=PRED(TRAITEMENT);
   PRED est le predecesseur d'un ensemble
   decrit explicitement comme MOTS_CLEF}
  ORTHOGRAPHE[TERME]:=MOT;
  while MOT<>ORTHOGRAPHE[TRAITEMENT] do
    TRAITEMENT:=PRED(TRAITEMENT);
  CODE_RESERVE:=TRAITEMENT
end;
procedure LIRECARACT(SITUATION:CONTEXTE;var CAR:char;
    var CAR_SPECIAL:boolean);

begin
  car:=car_suivant;
  if not eof(entree)
    then
      read(entree,car_suivant);
  CAR_SPECIAL:=car in carspe[situation];
  if (situation=programme)and (car='.')
    then
      car_special:=car_suivant=' '
end;

procedure liremot (situation:contexte;var dernier_car:char);

var
      car:char;
      car_ignores:set of char;
      car_special:boolean;

begin
  case situation of
    programme,parenthesee:
        car_ignores:=[' '];
    chaine:
        car_ignores:=[]
  end;
  tampon:='';
  repeat
    lirecaract(situation,car,car_special)
  until not (car in car_ignores)or eof(entree);
  while not car_special and not eof(entree)do
    begin
      memoriser(car);
      lirecaract(situation,car,car_special)
    end;
  dernier_car:=car
end;

procedure increment(i:integer);

var
      j:integer;

begin
  for j:=1 to i do
    if marge<margemax
      then
        marge:=marge+2
      else
        retard:=retard+1
end;

procedure decrement(i:integer);

var
      j:integer;

begin
  for j:=1 to i do
    if retard<>0
      then
        retard:=retard-1
      else
        marge:=marge-2
end;

procedure iniligne(marge:integer);

begin
  write(sortie,'':marge);
  nbcol:=marge
end;

procedure ecritmot;

var
      i:integer;

begin
  if nbcol+length(tampon)+1>=nbcolmax
    then
      begin
        increment(2);
        writeln(sortie);
        aff_traitement;
        iniligne(marge);
        decrement(2)
      end;
  nbcol:=nbcol+length(tampon);
  for i:=1 to length(tampon)do
    write(sortie,tampon[i])
end;
procedure empiler(i:integer;j:boolean);

begin
  pile_vide:=false;
  if sommet+1>longpile
    then
      erreur(2)
    else
      begin
        sommet:=sommet+1;
        with pile[sommet] do
          begin
            retour:=i;
            tabulation:=j
          end
      end
end;

procedure depiler(var a:integer);

begin
  if sommet=0
    then
      erreur(1)
    else
      begin
        a:=pile[sommet].retour;
        sommet:=sommet-1;
        pile_vide:=sommet=0
      end
end;
```

Fichier INDENT2.TEXT

```
{Fichier INDENT2.TEXT}


    procedure traitmot(traitement:MOTS_CLEF);

        procedure commentaire;

        begin
          repeat
            write(sortie,dernier_car);
            lirecaract(chaine,dernier_car,car_special)
          until dernier_car='}'
        end;

        procedure decal_terme;

        begin
          if changer_ligne
            then
              begin
                iniligne(marge);
                changer_ligne:=false
              end;
          ecritmot;
        end;

        procedure decal_begin;

        begin
          if sous_declar or sous_pro
            then
              begin
                writeln(sortie);
                aff_traitement;
                sommet:=0
              end;
          sous_pro:=false;
          if sous_declar
            then
              begin
                decrement(3);
```

```
        sous_declar:=false
      end;
  empiler(marge,true);
  iniligne(marge);
  ecritmot;
  increment(1);
  changer_ligne:=true;
  while car_suivant=' 'do
    lirecaract(programme,dernier_car,car_special);
  if car_suivant='{'
    then
      begin
        commentaire;
        write(sortie,dernier_car);
        lirecaract(programme,dernier_car,car_special)
      end
end;
procedure decal_case;

begin
  changer_ligne:=false;
  if sous_declar
    then
      begin
        empiler(marge,false);
        ecritmot
      end
    else
      begin
        iniligne(marge);
        ecritmot;
        empiler(marge,true);
        increment(1)
      end;
  sous_case:=true
end;

procedure decal_declarations;

begin
  writeln(sortie);
  aff_traitement;
  if sous_declar
    then
      decrement(3)
    else
      sous_declar:=true;
  iniligne(marge);
  ecritmot;
  increment(3);
  changer_ligne:=true;
  sous_pro:=false;
end;

procedure decal_do;

begin
  ecritmot;
  changer_ligne:=true
end;

procedure decal_conditionnelle;

begin
  writeln(sortie);
  aff_traitement;
  changer_ligne:=true;
  if traitement=else2
    then
      depiler(marge);
  iniligne(marge);
  empiler(marge,false);
  ecritmot;
  increment(1);
  sous_end:=false
end;

procedure decal_end;

    procedure decal_begin_end;

    begin
      if not changer_ligne
        then
          begin
            writeln(sortie);
            aff_traitement
          end;
      iniligne(marge);
      ecritmot;
      changer_ligne:=false;
      sous_end:=true;
      if pile_vide
        then
          begin
            niveau:=niveau-1;
            if 4*niveau<margemax
              then
                marge:=4*niveau
              else
                marge:=margemax
          end;
      if dernier_car=' '
        then
          begin
            while car_suivant=' 'do
              lirecaract(programme,dernier_car,car_special);
            if car_suivant='{'
              then
                begin
                  commentaire;
                  write(sortie,dernier_car);
                  lirecaract(programme,dernier_car,car_special)
                end
          end
    end;


    procedure decal_record_end;

    begin
      writeln(sortie);
      aff_traitement;
      iniligne(marge);
      ecritmot;
      if pile_vide
        then
          begin
            niveau:=niveau-1;
            if 2*niveau<margemax
              then
                marge:=2*niveau+6
              else
                marge:=margemax
          end
    end;
begin
  while not pile[sommet].tabulation do
    depiler(marge);
  depiler(marge);
  if sous_declar
    then
      decal_record_end
    else
      decal_begin_end
end;

procedure decal_boucle;

begin
  changer_ligne:=false;
  iniligne(marge);
  ecritmot;
  increment(1)
end;

procedure decal_sous_programme;

begin
  writeln(sortie);
  aff_traitement;
  if sous_declar
    then
      decrement(1)
    else
      increment(2);
  sous_declar:=false;
  changer_ligne:=false;
  iniligne(marge);
  niveau:=niveau+1;
  ecritmot;
  debut_pro:=true;
  sous_pro:=false
end;

procedure decal_of;

begin
  ecritmot;
  if sous_case
    then
      begin
        changer_ligne:=true;
        sous_case:=false
      end
end;

procedure decal_program;
```

```
        begin
          writeln(sortie);
          aff_traitement;
          debut_pro:=true;
          changer_ligne:=false;
          sous_pro:=false;
          marge:=0;
          niveau:=0;
          iniligne(marge);
          ecritmot
        end;

        procedure decal_record;

        begin
          empiler(nbcol,true);
          ecritmot;
          marge:=nbcol-6;
          increment(1);
          changer_ligne:=true;
          niveau:=niveau+1
        end;

        procedure decal_repeat;

        begin
          iniligne(marge);
          ecritmot;
          empiler(marge,true);
          increment(1);
          changer_ligne:=true
        end;

        procedure decal_until;

        begin
          if not changer_ligne
            then
              begin
                writeln(sortie);
                aff_traitement
              end;
          while not pile[sommet].tabulation do
            depiler(marge);
          depiler(marge);
          iniligne(marge);
          ecritmot;
          changer_ligne:=false
        end;
    begin (TRAITMOT)
      sous_end:=false;
      case traitement of
        terme:
            decal_terme;
        begin2:
            decal_begin;
        case2:
            decal_case;
        label2,const2,type2,var2:
            decal_declarations;
        do2:
            decal_do;
        then2,else2:
            decal_conditionnelle;
        end2:
            decal_end;
        for2,if2,with2,while2:
            decal_boucle;
        proc2,funct2:
            decal_sous_programme;
        of2:
            decal_of;
        program2:
            decal_program;
        record2:
            decal_record;
        repeat2:
            decal_repeat;
        until2:
            decal_until
      end
    end;
```

Fichier INDENT3.TEXT

```
(Fichier INDENT3.TEXT)


    procedure traitcarsort(var situation:contexte);

    var
      fin_ligne:boolean;
      i:integer;

    procedure traite_programme;

      procedure point;

      begin
        write(sortie,dernier_car);
        if sous_end or changer_ligne
          then
            begin
              writeln(sortie);
              aff_traitement
            end;
      end;

      procedure deux_points;

      begin
        if sous_declar or (car_suivant='=')
          then
          write(sortie,dernier_car)
        else
          begin
            write(sortie,dernier_car);
            changer_ligne:=not debut_pro;
            if not debut_pro
              then
                begin
                  writeln(sortie);
                  aff_traitement;
                  increment(2)
                end
          end
      end;

      procedure accogauche;

      begin
        if changer_ligne
          then
            iniligne(marge);
        write(sortie,dernier_car);
        repeat
          lirecaract(chaine,dernier_car,car_special);
          write(sortie,dernier_car)
        until dernier_car='}';
        writeln(sortie);
        aff_traitement;
        changer_ligne:=not sous_end
      end;

      procedure parengauche;

      begin
        if car_suivant='*'
          then
            begin
              fin_ligne:=false;
              writeln(sortie);
              aff_traitement;
              write(sortie,dernier_car);
              repeat
                lirecaract(chaine,dernier_car,car_special);
                write(sortie,dernier_car);
                if fin_ligne
                  then
                    begin
                      writeln(sortie);
                      aff_traitement
                    end;
                fin_ligne:=eoln(entree)
              until (dernier_car='*')and(car_suivant=')');
              lirecaract(chaine,dernier_car,car_special);
              writeln(sortie,dernier_car);
              aff_traitement
            end
          else
            begin
              situation:=parenthesee;
              write(sortie,dernier_car);
              nbrparentheses:=1;
              traitcarsort(situation)
            end
      end;

      procedure point_virgule;

      begin
        if not pile_vide and not sous_declar
          then
            begin
              while not pile[sommet].tabulation do
```

```
            depiler(marge);
          marge:=pile[sommet].retour;
          increment(1);
          writeln(sortie,dernier_car);
          aff_traitement;
          changer_ligne:=true;
          sous_end:=false
        end
      else
        if sous_declar
          then
            begin
              writeln(sortie,dernier_car);
              aff_traitement;
              changer_ligne:=true
            end
          else
            begin
              writeln(sortie,dernier_car);
              aff_traitement;
              if debut_pro
                then
                  begin
                    debut_pro:=false;
                        sous_pro:=true
                      end
                end;
        changer_ligne:=true
      end;

      procedure apostrophe;

      begin
        situation:=chaine;
        traitcarsort(situation);
        situation:=programme
      end;
  begin
    if dernier_car in [':','{','(',';','''','.']
      then
        case dernier_car of
          '.':
              point;
          ':':
              deux_points;
          '{':
              accogauche;
          '(':
              parengauche;
          ';':
              point_virgule;
          '''':
              apostrophe
        end
      else
        begin
          write(sortie,dernier_car);
          if changer_ligne
            then
              begin
                writeln(sortie);
                aff_traitement
              end
        end
end;

procedure traite_parenthese;

      procedure apostrophe;

      begin
        situation:=chaine;
        traitcarsort(situation);
        situation:=parenthesee;
        traitcarsort(situation)
      end;

      procedure parengauche;

      begin
        nbrparentheses:=nbrparentheses+1;
        write(sortie,dernier_car);
        traitcarsort(situation)
      end;

      procedure parendroite;

      begin
        write(sortie,dernier_car);
        nbrparentheses:=nbrparentheses-1;
        if nbrparentheses=0
          then
            situation:=programme
          else
                traitcarsort(situation)
            end;
        begin {traite_parenthese}
          liremot(parenthesee,dernier_car);
          if length(tampon)<>0
            then
              ecritmot;
          if dernier_car in['(','''',')']
            then
              case dernier_car of
                '''':
                    apostrophe;
                '(':
                    parengauche;
                ')':
                    parendroite
              end
            else
              begin
                write(sortie,dernier_car);
                traitcarsort(situation)
              end
        end;

        procedure traite_chaine;
        begin
          tampon:='';
          memoriser(dernier_car);
          repeat
            lirecaract(chaine,dernier_car,car_special);
            memoriser(dernier_car);
            if (dernier_car='''')and (car_suivant='''')
              then
                begin
                  lirecaract(chaine,dernier_car,car_special);
                  memoriser(dernier_car);
                  lirecaract(chaine,dernier_car,car_special);
                  memoriser(dernier_car)
                end
          until dernier_car='''';
          if changer_ligne
            then
              iniligne(marge);
          changer_ligne:=false;
          ecritmot
        end;
    begin
      nbcol:=nbcol+1;
      case situation of
        programme:
            traite_programme;
        parenthesee:
            traite_parenthese;
        chaine:
            traite_chaine
      end
    end;



begin {PROGRAMME PRINCIPAL}
  if fichiers_ouverts
  {Notez le type fonction booleenne en
   PASCAL qui permet d'effectuer une serie
   d'operations en effectuant un simple
   test..
   Il aurait ete plus lourd de declarer
   FICHIER_OUVERTS comme une procedure
   et de faire
         fichier_ouverts (var erreur:boolean)
         if not erreur
           ...........}
    then
      begin
        initialiser;
        aff_traitement;
        while not eof(entree)do
          begin
            liremot(situation,dernier_car);
            if length(tampon)>0
              then
                begin
                  transfert(mot_a_rechercher);
                  traitmot(code_reserve(mot_a_rechercher))
                end;
            traitcarsort(situation)
          end;
        close(entree,normal);
        close(sortie,lock);
        gotoxy(0,20);
        writeln(ligne:4,' lignes generees',' ':18)
      end
end.
```

Le château hanté

François-Xavier Nelva

Ce programme, écrit en Basic Étendu, est un jeu d'aventure en deux parties qui fonctionne sans extension de mémoire.

La première partie du programme redéfinit les caractères et, dans le même temps, se charge de la présentation.

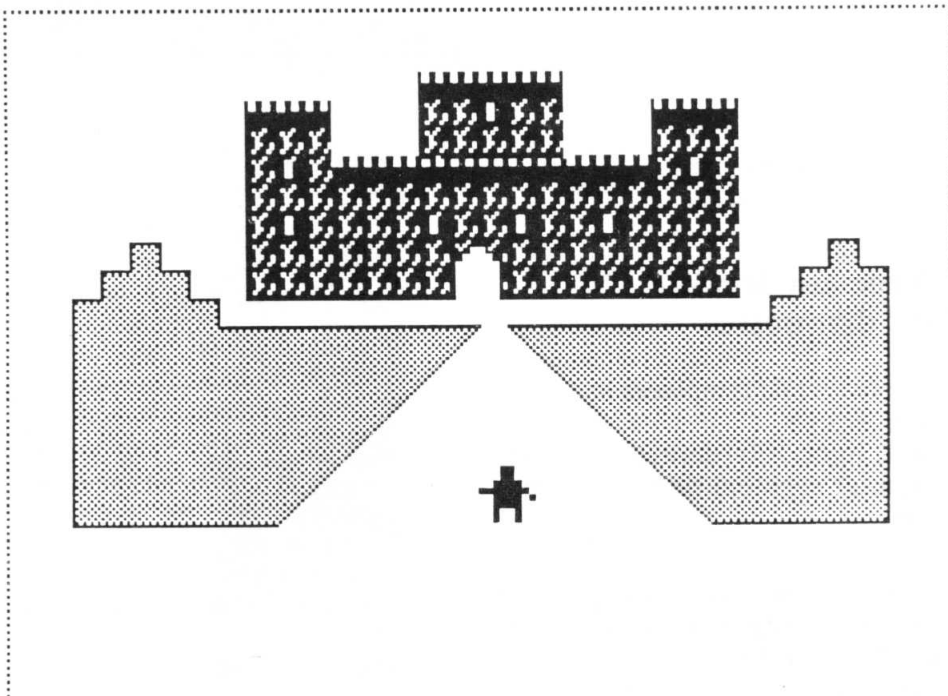
La seconde partie correspond au jeu lui-même.

On voit d'abord apparaître le plan du château (voir la copie d'écran). Vous devez, avec la manette de jeux numéro 1, retrouver le trésor caché. Vous avez au début 200 'points de force' qui diminuent au fur et à mesure que vous avancez (à zéro, vous êtes morts !). La première chose à faire, c'est de se diriger vers la tour numéro 1, où vous devrez combattre un dragon (il est impossible d'aller directement vers les autres tours car, dans ce cas, vous n'avez pas de bouclier, et alors...). Pendant le combat, vos forces déclinent mais vous gagnerez de toute façon.

Ensuite, vous devrez prendre le bouclier pour aller en direction de la porte A (en chemin, vous pouvez très bien tomber dans un souterrain, mais nous verrons ce 'détail' un peu plus loin).

Si tout c'est bien passé, vous arriverez dans un labyrinthe dont le tracé est invisible et où se déplace, aléatoirement, un fantôme (dangereux pour votre vie). Là, il vous faudra prendre l'épée et rejoindre rapidement la crypte par la porte située en haut et à gauche de l'écran.

Dans la crypte, on doit trouver la clé ainsi que le blason qui se trouve derrière une des quatre



portes indiquées sur la copie d'écran de la crypte. Pour pouvoir regarder derrière elles, il suffit de se placer devant ; vous réapparaissez de l'autre côté du mur. Il faut, bien entendu, faire attention aux deux fantômes. Si vous trouvez la clé, un code de un chiffre apparaîtra en bas et à gauche de l'écran. Ce code vous permettra d'ouvrir la porte derrière laquelle se trouve le trésor.

Cette porte existe à deux endroits (?!) : soit dans le plan du château (porte D), soit dans les souterrains.

Pour arriver à la porte du trésor (celle des souterrains), il faut d'abord passer par la crypte (porte indiquée). Ensuite, en faisant attention aux pierres, trouver le chemin caché de la porte du trésor. Là, on vous demandra le code ; si vous l'avez, vous gagnez.

Pour arriver à l'autre porte du trésor (celle du plan du château), il faudra retourner dans le labyrinthe et prendre la porte à droite. Vous réapparaîtrez au niveau de la porte B. vous devrez ensuite rejoindre la porte du trésor en faisant attention aux dragons. La porte C est une fausse porte qui ne mène nulle part ; elle est uniquement là pour que vous perdiez vos forces.

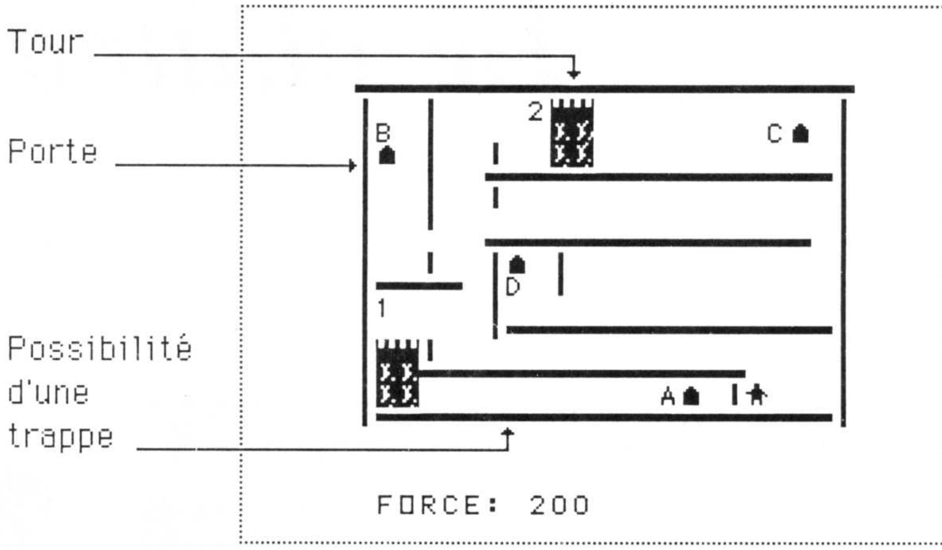
Pour conclure ce paragraphe, signalons que la clé, les boucliers, les dragons tués, l'épée et le blason rapporte des points pour les forces.

Chargement du programme

Ce jeu a été subdivisé en deux parties afin qu'il fonctionne sans extension de mémoire. Il faut d'abord charger la première

partie, puis RUN... Le programme affiche une image d'introduction, suivie des classiques messages de chargement depuis une cassette, ceci pour charger la seconde partie du programme. Ces messages sont relativement difficiles à lire du fait de la redéfinition de certains caractères, mais il s'agit bien de :

```
* REWIND CASSETTE TAPE     CS1
  THEN PRESS ENTER
* PRESS CASETTE PLAY       CS1
  THEN PRESS ENTER
* READING
* DATA OK
* PRESS CASETTE STOP       CS1
  THEN PRESS ENTER
```



Plan du Chateau

Avec un lecteur de disquettes

Il convient de remplacer le
`RUN "CS1"`
de la ligne 270 (dernière ligne du premier segment de programme) par
`RUN "DSK1.nom du segment"`
Ainsi, si vous avez sauvegardé le deuxième segment sous le nom de "CHATEAU2",
`RUN "DSK1.CHATEAU2"`
conviendra parfaitement.

99

Programme "Chateau hanté"

***Note** : afin d'en faciliter le comptage, les espaces situés dans les chaînes de caractères sont ici visualisés à l'aide de 'puces' (•).*

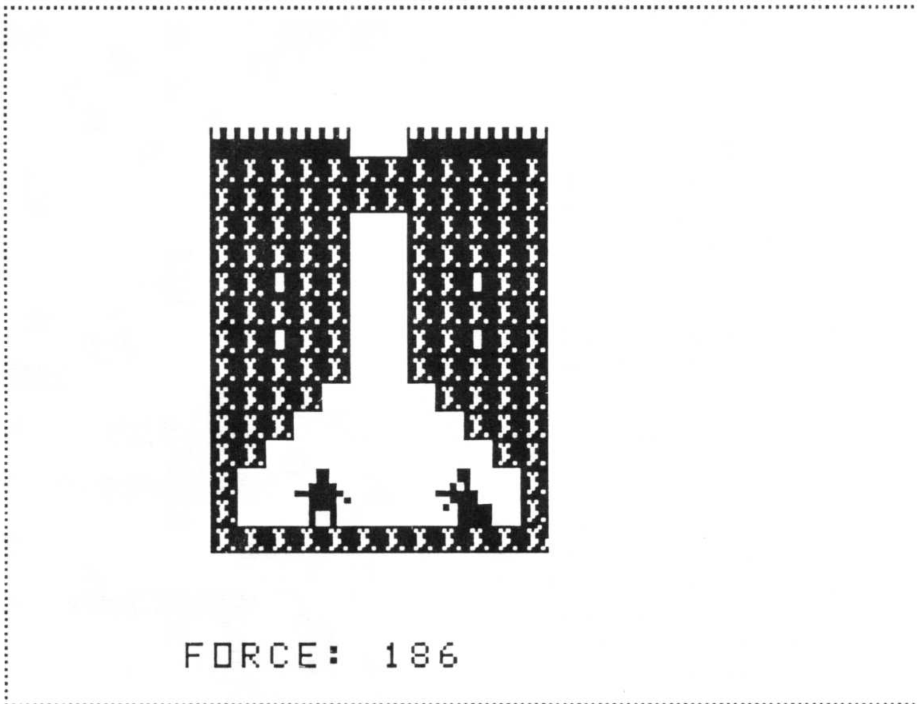
Première partie

Basic Étendu

```
10 CALL CLEAR
20 DATA 42,999999FFFFFFFF
   FF,59,0428101019112200
   ,43,18183CFE3D3C2424,6
   4,183C5A7FFFFFF3FF,120
   ,00183C7E7E7E7E7E00
30 DATA 44,000000C0E0C1FD
   3F,128,00041A2D3755AED
   D,140,181828FC3C5F1F1F
   ,129,0000E0BFE7010000,
   130,00002030FF302000,1
   41,007C7C6C446C7C38
31 DATA 97,FFE7E7E7E7E7FF
   FF,115,FFFEFCF8F0E0C08
   0,116,FF7F3F1F0F070301
   ,104,000000FFFFFF0000,
   105,1818181818181818
35 DATA 46,3C7EFFFFFFFF7E
   3C,106,FFFFFCF8F0F0E0E
   0,107,FFFF3F1F0F0F0707
   ,45,1818FFFF18181818
40 FOR I=1 TO 20
50 READ A,B$
60 CALL CHAR(A,B$)
70 CALL HCHAR(3+I,I+2,A)
80 NEXT I
90 FOR I=1 TO 500 :: NEXT
   I
100 CALL COLOR(11,13,2)::
    CALL SCREEN(2)
110 CALL COLOR(2,14,2)::
   CALL COLOR(4,2,14):: C
   ALL COLOR(5,14,2):: CA
   LL COLOR(12,14,7)
120 CALL COLOR(13,11,2)::
    CALL COLOR(14,9,2)::
   CALL COLOR(9,14,2)
130 CALL COLOR(10,13,13):
   : CALL CLEAR
140 A$="•••••••••••••••••••
   ••••••••••••••••••••••••
   ••••••••••••••••••••••••
   •••••••*****••••••••••••
   •••••••***•••;;a;;•••*
   **•••••"
150 B$="••••••;;;•••;;;;;
   •••;;;•••••••••••••;a;**
   **********;a;••••••••••
   •;;;;;;;;;;;;;;;;;;;••••
   "
160 C$="••••••;a;;;;a;;a;
   ;a;;;•••••••h•••;;;;;
   ;;;;;;;;;;;;;;•••h••hhh•
   •;;;;;;;;;H;;;;;;;;;••hh
   h"
165 CALL COLOR(10,13,13)
170 D$="hhhhh•••••••••••••
   •••••••hhhhhhhhhhhhhhh
   hhs•thhhhhhhhhhhhhhhhhh
   hhhhhhhs•••thhhhhhhhhh
   hhhhhhhhhhhhs•••••thhh
   hhhhhhh"
172 E$="hhhhhhhhhhs••••••
   •thhhhhhhhhhhhhhhhhhs•
   ••••••••thhhhhhhhhhhhh
   hhhs•••••••••••thhhhhh
   hhhhhhhhs••••••••••••••
   thhhhhh"
180 DISPLAY AT(1,1):A$;B$
   ;C$;D$;E$
190 CALL SPRITE(#1,120,2,
   65,126):: CALL SPRITE(
   #3,43,5,130,128)
200 CALL MAGNIFY(2)
215 FOR I=1 TO 2
220 CALL SOUND(500,196,0)
   :: CALL SOUND(500,196,
   2):: CALL SOUND(300,17
   5,0):: CALL SOUND(500,
   196,0):: CALL SOUND(30
   0,220,0)
230 CALL SOUND(300,233,0)
   :: CALL SOUND(500,220,
   0):: CALL SOUND(500,22
   0,2)
240 CALL SOUND(500,196,0)
   :: CALL SOUND(500,175,
   0):: CALL SOUND(500,19
   6,0):: NEXT I
250 CALL SOUND(30,380,0):
   : CALL MOTION(#3,-5,0)
   :: CALL COINC(#3,#1,3,
   C):: IF C=-1 THEN 260
   ELSE 250
260 CALL MOTION(#3,0,0)::
    FOR I=3 TO 16 :: CALL
   POSITION(#1,W,WW):: C
   ALL SPRITE(#1,43,I,W,W
   W):: CALL SOUND(200,-5
   ,0,110,0):: NEXT I
270 CALL DELSPRITE(#3,#1)
   :: RUN "CS1"
```

***Note** : si vous utilisez un lecteur de*

disquettes, remplacez RUN "CS1" par RUN "DSK1.CHATEAU2" (si, bien sûr, la seconde partie a été sauvegardée sous le nom de "CHATEAU2").



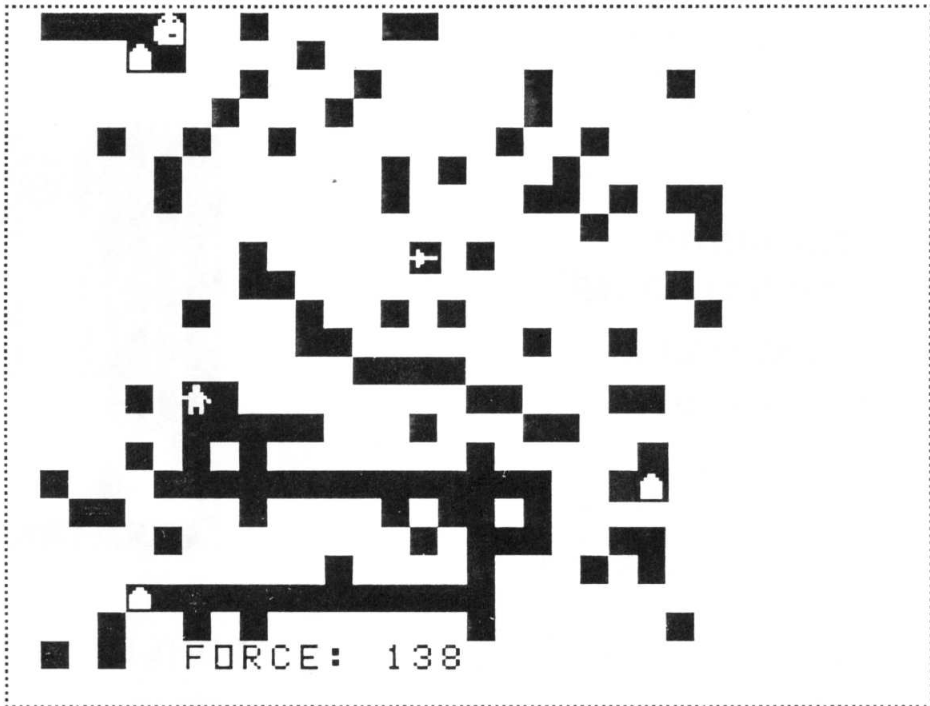
Programme "Chateau hanté"

Seconde partie

Basic Étendu

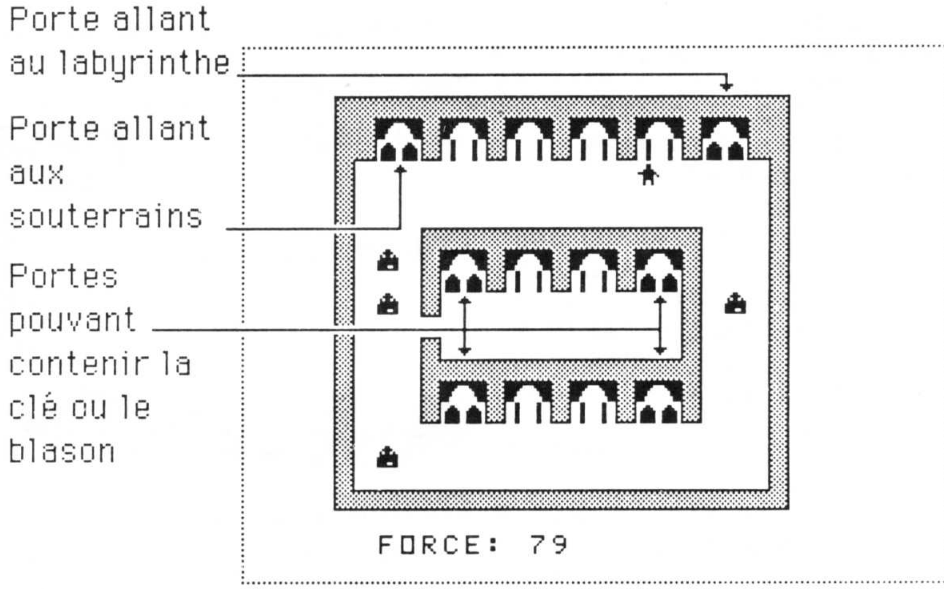
```
1 RZ=64 :: RR=0 :: RANDOM
    IZE :: Z=1 :: MN=0
2 VV=INT(RND*9)+1 :: N=18
    :: M=24 :: FO=200
10 DIM P$(25)
250 CALL CLEAR :: CALL SC
    REEN(16):: CALL COLOR(
    2,14,16):: CALL COLOR(
    4,2,14):: CALL COLOR(1
    0,2,16):: CALL COLOR(1
    2,9,16)
251 CALL DELSPRITE(#1)
260 AA$="•••hhhhhhhhhhhhh
    hhhhhhhhhh•••••i••i•••
    ••**•••••••••••••i•••••i
    ••i••••••;;••••••••••x•i
    •••••ix•i••i••;;••••••
    •••••i••"
270 AB$="•••i••i••hhhhhhh
    hhhhhhhhhi•••••i••i••i
    ••••••••••••••••i•••••i
    ••i••••••••••••••••••••i
    •••••i•••••hhhhhhhhhhh
    hhhh•i••"
280 AC$="•••i••i••ix•i•••
    ••••••••••i•••••ihhhh•i
    ••i••••••••••••••i•••••i
    •••••i••••••••••••••••i
    •••••i•••••ihhhhhhhhhh
    hhhhhi••"
290 AD$="•••i**i•••••••••
    ••••••••••i•••••i;;hhhh
    hhhhhhhhhhh••••i•••••i
    ;;••••••••••••x•i••••i
    •••••ihhhhhhhhhhhhhhhh
    hhhhhi"
300 DISPLAY AT(4,1):AA$:A
    B$:AC$:AD$ :: CALL HCH
    AR(N,M,43)
305 CALL MAGNIFY(1)
310 GOSUB 5000 :: T=INT(R
    ND*4)+1
320 CALL JOYST(1,X,Y)
325 IF X=0 AND Y=0 THEN 3
    20
330 B=M :: A=N
340 M=X*0.25+M :: N=-Y*0.
    25+N
350 CALL GCHAR(N,M,C)
360 IF C=104 OR C=105 THE
    N 600
370 IF C=42 OR C=59 THEN
    1000
375 IF M=11 AND N=13 AND
    Z<2 THEN 6000
378 IF M=27 AND N=13 THEN
    800
379 IF M=14 AND N=13 THEN
    720
380 FO=FO-1 :: GOSUB 5000
400 CALL HCHAR(N,M,43)::
    CALL HCHAR(A,B,32)
405 IF M=21 AND N=18 THEN
    700
410 IF M=13 AND N=18 AND
    T=2 THEN 420 ELSE 320
420 CALL HCHAR(18,13,30,1
    ):: CALL SOUND(500,176
    0,0):: GOTO 8500
600 M=B :: N=A
610 GOTO 320
700 M=7 :: N=21 :: GOTO 3
    000 :: FO=200
720 DISPLAY AT(24,1):"COD
    E:" :: ACCEPT AT(24,6)
    :PO
730 IF PO=VV THEN 2810 EL
    SE 6000
800 IF Z>1 THEN V=50 ELSE
    V=0
810 FO=FO-1 :: GOSUB 5000
    :: IF FO<V THEN 250
820 CALL HCHAR(N,M-1,43):
    : CALL HCHAR(N,20,140)
    :: CALL HCHAR(N,M-1,32
    ):: CALL HCHAR(N,20,32
    )
830 CALL HCHAR(N,M-3,43):
    : CALL HCHAR(N,21,140)
    :: CALL HCHAR(N,M-3,32
    ):: CALL HCHAR(N,21,32
    )
840 CALL SOUND(30,380,0):
    : Z=Z-1 :: GOTO 810
1000 CALL CLEAR :: V=INT(
    RND*(150-80))+80
1110 BA$="•••••******••***
    **••••••••••••••••;;;;
    ;;;;;;;;••••••••••••••
    ••;;;;;;;;;;;;•••••••
    •••••••;;;;;••;;;;;••
    ••••••••"
1130 BC$="•••••;;;;;••;;;
    ;;•••••••••••••••••;;a;
    ;••;;a;;•••••••••••••••
    ••;;;;;••;;;;;•••••••
    ••••••••;;a;;••;;a;;••
    ••••••••"
1140 BD$="•••••;;;;;••;;;
    ;;•••••••••••••••••;;;;
    ••••;;;;•••••••••••••••
    ••;;;•••••••;;;•••••••
    ••••••••;;•••••••••;;••
    ••••••••"
1150 BE$="•••••;••••••••••
    •;•••••••••••••••••;•••
    •••••••;••••••••••••••
    ••;;;;;;;;;;;;;•••••••
    •••"
1155 DISPLAY AT(5,1):BA$;
    BC$;BD$;BE$
1160 CALL MAGNIFY(2)
1180 CALL SPRITE(#3,141,9
    ,25,97)
1190 CALL SOUND(30,380,0,
    -5,0)
1200 FO=FO-1 :: GOSUB 500
    0 :: IF FO<V THEN 1300
1210 CALL SPRITE(#1,43,2,
    130,90,#2,140,9,130,10
    0):: CALL SPRITE(#1,43
    ,2,130,80,#2,140,9,130
    ,110)
1220 GOTO 1190
1300 FO=FO+40 :: GOSUB 50
```

```
   00 :: FOR I=1 TO 5 ::
   CALL SOUND(40,380,0)::
    NEXT I :: Z=Z+1
1310 CALL JOYST(1,X,Y)::
   CALL MOTION(#1,-Y,X)::
    CALL DELSPRITE(#2)
1320 CALL COINC(#1,#3,5,C
   ):: IF C=-1 THEN 1330
   ELSE GOTO 1310
1330 FO=FO+30 :: GOSUB 50
   00 :: CALL MOTION(#1,0
   ,0):: CALL DELSPRITE(#
   3)
1350 M=9 :: N=18 :: GOTO
   250
1500 CALL COLOR(11,6,6)
1510 P$(9)="•••wwwwwwwwww
   wwwwwwwwwww•••••••••wwww
   wwwwwwwwwwwwwwwww••••••
   ••wwwwwwwwwwwwwwwwwwwwww
   ••••••••••••••w•••••••••
   •••••••••••••"
1520 P$(10)="••••••••••w••w
   wwwwwwwww••!••••••••••••
   ••w••xxxxxxxxx••!•••••
   ••••••••••w••aaaaaaaaa!!
   !••••••••••••••w••••••••
   ••••••!!••••"
1530 P$(11)="••••••••www••
   •••••••••••!••••••••••••
   wwwww••••••••x!••!•••••
   •••••wwwwwww•••••••••!•!
   !••••••••••••••w••••••••
   •••!••!•••••"
1540 P$(12)="•••wwwwwwwwww
   w•••••••!••!••••••••••www
   wwwwwww••••••!••!•••••
   •••••••••!•••••••••••!••
   !!•••••••••••••••!••••••
   !!!!••!•••••"
1550 P$(13)="•••••••••••!••
   ••••!•••••!•••••••••••••
   •!!!!!!!!!!!!!!!!!!!!!••••
   "
1560 CALL SCREEN(2)
1570 DISPLAY AT(4,1):P$(9
   ):P$(10):P$(11):P$(12)
   :P$(13)
1580 FOR I=3 TO 7 :: CALL
    COLOR(I,16,2):: NEXT
   I
1700 CALL HCHAR(N,M,43)::
    Q=25 :: L=1 :: QQ=8
1800 MZ=6 :: NZ=17
1900 CC=1
2000 FO=FO-1 :: GOSUB 500
   0
2005 BZ=MZ :: AZ=NZ
2006 AL=Q :: LL=QQ
2010 CALL JOYST(1,S,D)::
   MZ=MZ+CC :: QQ=QQ+L ::
    CALL GCHAR(NZ,MZ,Y)::
    CALL GCHAR(QQ,Q,R)::
   IF Y=32 THEN 2500 :: I
   F Y=43 THEN 6000
2011 IF R=32 THEN 2600 ::
```



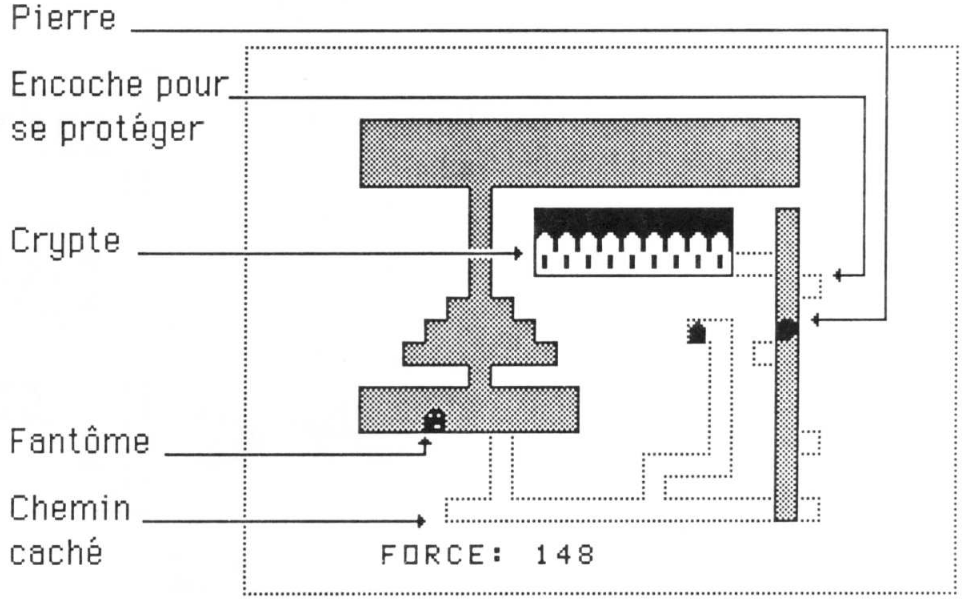

```
   IF R=43 THEN 6000 ::
   CALL HCHAR(AZ,BZ,119):
   : CALL HCHAR(NZ,MZ,64)
2012 CALL HCHAR(LL,AL,119
   ):: CALL HCHAR(QQ,Q,46
   )
2015 B=M :: A=N
2020 M=S*0.25+M
2030 N=-D*0.25+N
2040 CALL GCHAR(N,M,X)
2050 IF X=32 THEN 2300
2060 IF X=64 THEN 6000
2061 IF X=120 THEN 2350
2065 IF N=14 AND M=22 THE
   N 2800
2070 IF FO=0 THEN 6000
2090 CALL HCHAR(A,B,119):
   : CALL HCHAR(N,M,43)
2100 GOTO 2000
2300 M=B :: N=A :: BZ=MZ
   :: AZ=NZ :: MZ=MZ+CC :
   : CALL GCHAR(NZ,MZ,Y):
   : IF Y=32 THEN 2500 ::
    CALL HCHAR(AZ,BZ,119)
2310 IF Y=43 THEN 6000 ::
    LL=QQ :: AL=Q :: QQ=Q
   Q+L :: CALL GCHAR(QQ,Q
   ,R):: IF R=32 THEN 260
   0 :: CALL HCHAR(LL,AL,
   119):: GOTO 2010
2350 M=7 :: N=7 :: GOTO 4
   000
2500 CC=-CC :: CALL HCHAR
   (AZ,BZ,64):: GOTO 2010
2600 QQ=LL :: Q=AL :: CAL
   L HCHAR(QQ,Q,119):: QQ
   =8 :: GOTO 2010
2800 DISPLAY AT(24,1)BEEP
   :"DONNE LE CODE:" :: A
   CCEPT AT(24,15):AS ::
   IF AS=VV THEN 2810 ELS
   E 2830
2810 CALL CLEAR :: CALL M
   AGNIFY(2):: CALL SPRIT
   E(#8,128,12,130,100,#1
   ,43,5,130,120):: FOR F
   =1 TO 3
2811 FOR I=1 TO 16 :: CAL
   L SCREEN(I):: CALL SOU
   ND(100,(I*7)+110,0)::
   DISPLAY AT(20,12):"GAG
   NE!!!!!!" :: NEXT I
2820 NEXT F :: GOTO 7000
2830 M=BZ :: N=AZ :: GOTO
    2010
3000 P$(14)="••xZZZZZZZZZ
   ZZZZZZZZ•••••••••••Z•
   ••Z••••••••Z•Z•••••••
   ••Z•Z•Z•Z•ZZZZZZZZZ•Z•
   ••••••••Z•ZZZ•Z••Z••••
   ••Z•Z••••••••"
3010 P$(15)="••Z•Z•••Z•ZZ
   ZZZ•Z•Z•Z•••••••••ZZZZ
   Z•Z••••Z••Z•Z•Z•••••••
   ••Z•••••Z•ZZZZZZZ•ZZZ•
   ••••••••ZZ•ZZZZ•ZZZZZ•
   Z•••••••••••"
3020 P$(16)="••Z••Z••••••
   ••••ZZZZZ•••••••••Z•ZZ
   Z•Z•ZZZZ•ZZ•Z•Z•••••••
   ••Z•Z•Z•Z••Z•Z••Z•Z•Z•
   ••••••••ZZZ•ZZZZZZ•ZZZ
   Z•Z•Z••••••••"
3030 P$(17)="••••Z•••••••
   •••••••••••••••••••Z•ZZ
   ZZZZZZZZZZZZZZ•••••••
   ••Z•••Z••••••••••••Z•Z•
   ••••••••ZZZZZZZZZZZZZ
   Z•Z•x••••••••"
3040 P$(18)="••Z•••••••••
   Z•••••Z•Z••••••••••Z•ZZ
   ZZZZZZZ•ZZZ•Z•Z•••••••
   ••Z••••••••••Z•Z•Z•Z•••
   •••••••••xZZZZZZZZZZZ•
```

```
      ZZZZZ•••••••"
3043 FOR I=3 TO 7 :: CALL
      COLOR(I,16,2):: NEXT
      I
3050 ÇALL SCREEN(2):: CAL
     L COLOR(8,2,2):: CALL
     COLOR(11,6,6)
3060 DISPLAY AT(1,3):P$(1
     4):P$(15):P$(16):P$(17
     ):P$(18)
3070 IF MN<1 THEN CALL HC
     HAR(9,15,130):: CALL C
     OLOR(13,12,2)
3075 MZ=12 :: NZ=12
3080 CALL JOYST(1,X,Y)
3085 A=N :: B=M
3090 BZ=MZ :: AZ=NZ
3100 N=-Y*0.25+N :: M=X*0
     .25+M
3110 MZ=INT(RND*25+1):: N
     Z=INT(RND*23+1):: CALL
      GCHAR(NZ,MZ,SS):: IF
     SS=120 OR SS=130 THEN
     MZ=MZ+1
3120 CALL HCHAR(AZ,BZ,119
     ):: CALL HCHAR(NZ,MZ,6
     4)
3130 CALL COLOR(2,9,6)::
     CALL GCHAR(N,M,XX):: I
     F XX=32 THEN 3300 :: I
     F XX=130 THEN 3400 ::
     IF XX=64 THEN 6000
3140 FO=FO-1 :: GOSUB 500
     0
3150 IF M=5 AND N=2 THEN
     3720 :: IF M=5 AND N=2
     1 THEN 3700 :: IF M=23
      AND N=17 THEN 3750
3160 IF FO<0 THEN 6000
3170 CALL HCHAR(A,B,119):
     : CALL HCHAR(N,M,43)
3180 GOTO 3080
3300 M=B :: N=A :: GOTO 3
     080
3400 FO=FO+50 :: GOSUB 50
     00 :: CALL SOUND(100,3
     80,0,1760,0):: MN=1 ::
      M=B :: N=A :: CALL HC
     HAR(9,15,119):: GOTO 3
     080
3700 M=18 :: N=18 :: GOTO
      250
3720 M=21 :: N=6 :: GOTO
     4000
3750 M=8 :: N=9 :: GOTO 2
     50
4000 CALL CLEAR :: CALL S
     CREEN(6):: CALL HCHAR(
     N,M,43):: GG=INT(RND*2
     )+1
4010 CALL COLOR(9,5,5)
4020 P$(19)="••cccccccccc
     ccccccccccc•••••••ccjk
     cjkcjkcjkcjkcjkcc•••••
     ••ccxxciiciiciiciicxxc
     c•••••••c•••••••••••••
     ••••••c••••"
4030 P$(20)="••c•••••••••
     ••••••••••c•••••••c•••
     ••••••••••••••••c•••••
     ••c•••ccccccccccccc•••
     c•••••••c•••cjkcjkcjkc
     jkc•••c••••••"
4040 P$(21)="••c•••cxxcii
     ciicxxc•••c•••••••c•••
     c•••••••••••c•••c•••••
     ••c•••••••••••••••c•••
     c•••••••c•••c•••••••••
     ••c•••c••••••"
4050 P$(22)="••c•••cccccc
     ccccccc•••c•••••••c•••
     cjkcjkcjkcjkc•••c•••••
     ••c•••cxxciiciicxxc•••
     c•••••••c•••••••••••••
     ••••••c••••••"
4060 P$(23)="••c•••••••••
     ••••••••••c•••••••c•••
     ••••••••••••••••c•••••
     ••cccccccccccccccccccc
     c•••••"
4090 DISPLAY AT(3,1):P$(1
     9):P$(20):P$(21):P$(22
     ):P$(23)
4100 H=7 :: P=16 :: U=23
     :: E=8
4105 BB=1 :: DD=1
4110 CALL SOUND(100,-5,0,
     110,0)
4120 CALL JOYST(1,X,Y)
4130 HZ=H :: PZ=P :: UZ=U
      :: EZ=E
4140 P=P-BB :: E=E+DD ::
     CALL GCHAR(P,H,TZ):: C
     ALL GCHAR(E,U,ZT)
4150 IF TZ=43 AND RR=0 TH
     EN 6000 :: IF ZT=43 AN
     D RR=0 THEN 6000 :: IF
      TZ=43 AND RR=1 THEN 4
     950 :: IF ZT=43 AND RR
     =1 THEN 4950
4155 FO=FO-1 :: GOSUB 500
     0 :: IF FO<0 THEN 6000
4160 IF TZ=120 OR TZ=99 T
     HEN 4400 :: IF ZT=120
     OR ZT=99 THEN 4420
4170 CALL HCHAR(PZ,HZ,32)
      :: CALL HCHAR(EZ,UZ,32
     ):: CALL HCHAR(P,H,RZ)
      :: CALL HCHAR(E,U,RZ)
4180 B=M :: A=N :: M=X*0.
     5+M :: N=-Y*0.25+N
4190 CALL GCHAR(N,M,XX)::
      IF XX=64 AND RR=0 THE
     N 6000 :: IF XX=99 THE
     N 4450
4194 IF X=0 AND Y=0 THEN
     4120
4200 IF M=7 AND N=6 THEN
     4500 :: IF M=23 AND N=
     6 THEN 4550 :: IF M=11
      AND N=18 THEN 4600 ::
      IF M=19 AND N=18 THEN
      4650
4210 IF M=11 AND N=12 THE
     N 4900 :: IF M=19 AND
     N=12 THEN 4970
4220 CALL HCHAR(A,B,32)::
      CALL HCHAR(N,M,43)
4230 GOTO 4120
4400 P=PZ :: H=HZ :: BB=-
     BB :: GOTO 4120
4420 E=EZ :: U=UZ :: DD=-
     DD :: GOTO 4120
4450 M=B :: N=A :: GOTO 4
     120
4500 M=24 :: N=10 :: GOTO
      1500
4550 M=6 :: N=3 :: GOTO 3
     000
4600 IF GG=2 THEN 4610 EL
```

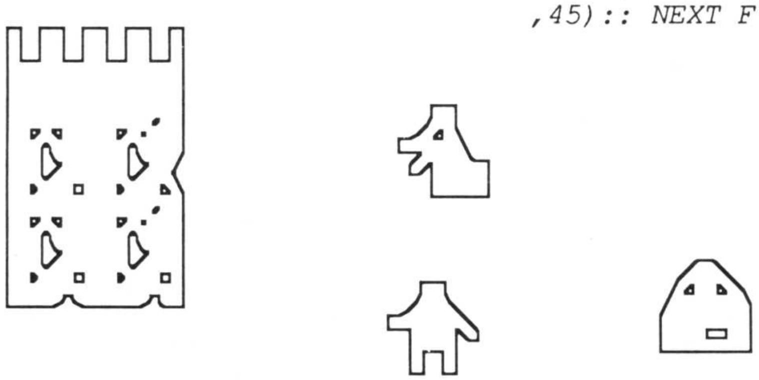


Crypte

```
   SE 4602
4602 M=B :: N=A :: CALL H
   CHAR(N,M,32):: CALL SO
   UND(100,-5,0,110,0)::
   N=13 :: GOTO 4120
4610 CALL HCHAR(17,11,141
   ):: CALL SOUND(1000,52
   9,0):: FOR I=1 TO 500
   :: NEXT I :: CALL HCHA
   R(17,11,120):: N=A ::
   M=B :: CALL HCHAR(N,M,
   32)
4611 RR=1 :: N=13 :: GOTO
    4120
4650 IF GG=1 THEN 4660 EL
   SE 4651
4651 N=A :: M=B :: CALL H
   CHAR(N,M,32):: CALL SO
   UND(100,-5,0,110,0)::
   N=14 :: GOTO 4120
4660 RR=1 :: CALL HCHAR(1
   7,19,141):: CALL SOUND
   (1000,529,0):: FOR I=1
    TO 500 :: NEXT I :: C
   ALL HCHAR(17,19,120)::
    M=B :: N=A
4661 CALL HCHAR(N,M,32)::
    N=14 :: GOTO 4120
4900 IF GG=1 THEN 4910
4905 N=A :: M=B :: CALL H
   CHAR(N,M,32):: CALL SO
   UND(100,-5,0,110,0)::
   N=8 :: GOTO 4120
4910 CALL HCHAR(11,11,129
   ):: CALL SOUND(1000,52
   9,0):: DISPLAY AT(24,1
   ):"CODE=";VV :: FOR I=
   1 TO 500 :: NEXT I ::
```



Souterrains



```
   M=B :: N=A :: CALL HCH
   AR(N,M,32)
4911 N=8 :: GOTO 4120
4950 FO=FO+20 :: GOSUB 50
   00
4955 RZ=32 :: GOTO 4120
4970 IF GG=2 THEN 4980 EL
   SE 4975
4975 M=B :: N=A :: CALL H
   CHAR(N,M,32):: CALL SO
   UND(100,-5,0,110,0)::
   N=8 :: GOTO 4120
4980 CALL HCHAR(11,19,129
   ):: CALL SOUND(1000,52
   9,0):: FOR I=1 TO 500
   :: NEXT I :: M=B :: N=
   A :: CALL HCHAR(N,M,32
   ):: N=8
4985 DISPLAY AT(24,1):"CO
   DE=";VV :: GOTO 4120
5000 DISPLAY AT(23,5)SIZE
   (11):"FORCE:";FO :: RE
   TURN
6000 CALL SCREEN(2):: CAL
   L CLEAR :: FOR F=1 TO
   10 :: CALL HCHAR(INT(R
   ND*20)+1,INT(RND*30)+1
   ,45):: NEXT F
6010 CALL MAGNIFY(2):: CA
   LL COLOR(2,9,2):: CALL
    SPRITE(#1,44,5,125,90
   )
6020 CALL SOUND(1000,523,
   0,659,0):: CALL SOUND(
   1000,392,0,523,0):: CA
   LL SOUND(1000,330,0,39
   2,0):: CALL SOUND(1000
   ,262,0,330,0)
6030 CALL SOUND(1000,131,
   0)
6040 G=G+1 :: IF G=4 THEN
    7000
6050 M=24 :: N=18 :: GOTO
    1
7000 CALL CLEAR :: DISPLA
   Y AT(10,10):"ENCORE 1
   FOIS?:" :: ACCEPT AT(1
   0,26):S$
7010 IF S$="o" THEN 1 ELS
   E END
8500 M=23 :: N=5 :: GOTO
   1500
```

Aux Texans d'Alsace

Jusque là, "99 Magazine" constituait le lien entre les utilisateurs encore mordus par les qualités du TI-99/4A. Ce dernier lien disparaît. Pour ceux qui ne tiennent pas à tout ranger de suite dans un coin du grenier, pourquoi ne pas garder le contact. Mon nom :

Henri Mathian
76 bis, rue de la Vallée
67140 BARR
Tél. : 88.08.16.14

Dessins en Basic TI

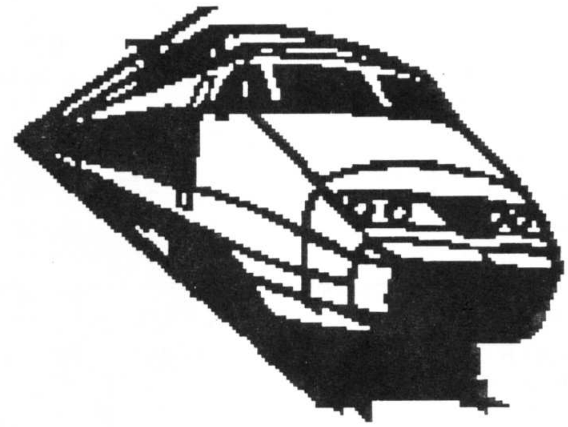
Jean-Pierre Dhée

Ceci n'est pas à proprement parler un programme ; il s'agit plutôt d'une 'fantaisie' graphique visant à représenter notre Train à Grande Vitesse national sur l'écran du TI, en Basic TI.

Ce travail a été réalisé par les élèves dessinateurs du Lycée Professionnel d'Ermont - Val d'Oise - sous la direction de Jean-Pierre Dhée, leur professeur.

La seconde copie d'écran représentant une maison en perspective (nous n'avons pas joint le listing, faute d'avoir assez de place) est issue de la même équipe.

99



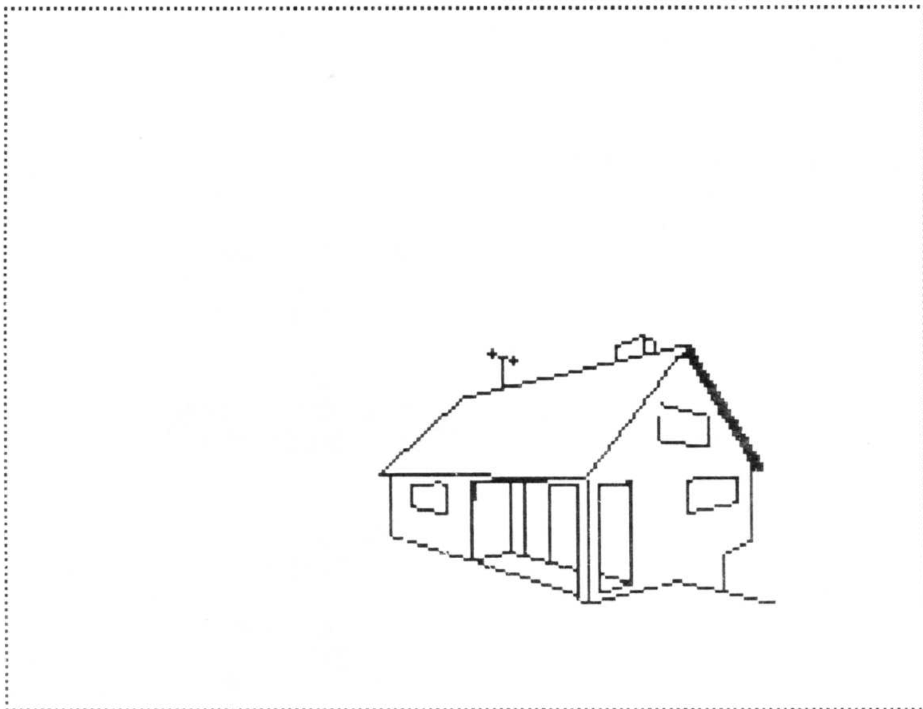
Programme "Dessin du TGV"

Basic TI

```
10 REM
20 REM
30 REM
40 REM      ****************
   ***
50 REM
60 REM      * LYCEE D'ERMONT
   *
70 REM        DHEE Jean Pier
   re
80 REM      ****************
   ***
90 REM
100 REM
120 REM
130 REM
140 REM
150 CALL SCREEN(3)
160 CALL CLEAR
170 CALL CHAR(159,"FFFFFF
   FFFFFFFFFF")
180 CALL HCHAR(12,10,159)
190 CALL HCHAR(8,17,159)
200 CALL HCHAR(9,21,159)
210 CALL HCHAR(10,8,159,6
   )
220 CALL HCHAR(13,23,159)
230 CALL HCHAR(11,12,159)
240 CALL HCHAR(14,13,159,
   2)
250 CALL HCHAR(15,14,159)
260 CALL HCHAR(16,15,159)
270 CALL HCHAR(15,21,159,
   4)
280 CALL HCHAR(16,21,159,
   5)
290 CALL HCHAR(17,16,159,
   2)
300 CALL HCHAR(18,17,159,
   6)
310 CALL CHAR(158,"C0FFFF
   FFFFFFFFFF")
320 CALL HCHAR(17,18,158)
330 CALL CHAR(34,"C0C0808
   080000000")
340 CALL HCHAR(16,26,34)
350 CALL CHAR(35,"0F81FFF
   FFFFFFFFF")
360 CALL HCHAR(17,19,35)
370 CALL CHAR(36,"FFFFFFF
   FFFFFF8FC")
380 CALL HCHAR(17,23,36)
390 CALL CHAR(37,"0000000
   000000107")
400 CALL HCHAR(6,12,37)
410 CALL CHAR(38,"01030F1
   F7EF8F0C3")
420 CALL HCHAR(6,13,38)
430 CALL CHAR(39,"F8E0C00
   0003FFFFF")
440 CALL HCHAR(6,14,39)
450 CALL CHAR(40,"0000000
   000FFFFFF")
460 CALL HCHAR(6,15,40,2)
470 CALL CHAR(41,"0000000
   00000FCFF")
480 CALL HCHAR(6,17,41)
490 CALL CHAR(42,"001F1F1
   3030F3F7C")
500 CALL HCHAR(7,11,42)
510 CALL CHAR(43,"1FFFFFF
   9FBFFFFFF")
520 CALL HCHAR(7,12,43)
530 CALL CHAR(44,"871FFFF
   FFFFFFFFF")
540 CALL HCHAR(7,13,44)
550 CALL CHAR(45,"FFFEFEF
   EEFEFFFFF")
560 CALL HCHAR(7,14,45)
```

```
570 CALL CHAR(46,"60001F7FFFFFFECC")
580 CALL HCHAR(7,15,46)
590 CALL CHAR(47,"0300FFFF8FFF0300")
600 CALL HCHAR(7,16,47)
610 CALL CHAR(48,"FF07FFFF00FFFF00")
620 CALL HCHAR(7,17,48)
630 CALL CHAR(49,"F8FFFFFF03FCFF00")
640 CALL HCHAR(7,18,49)
650 CALL CHAR(50,"0080F0FEFF3FFBFF")
660 CALL HCHAR(7,19,50)
670 CALL CHAR(51,"00000000C0F0FEFF")
680 CALL HCHAR(7,20,51)
690 CALL CHAR(52,"0000000000000107")
700 CALL HCHAR(8,9,52)
710 CALL CHAR(53,"0003071F3EF9F3CF")
720 CALL HCHAR(8,10,53)
730 CALL CHAR(54,"FEE3CF1F7FFBE79F")
740 CALL HCHAR(8,11,54)
750 CALL CHAR(55,"FFFFFFFFFFFFFCF0")
760 CALL HCHAR(8,12,55)
770 CALL CHAR(56,"FFFFFFFEE0810303")
780 CALL HCHAR(8,13,56)
790 CALL CHAR(57,"FFF8C0F9F3B3B727")
800 CALL HCHAR(8,14,57)
810 CALL CHAR(58,"7F7FFFFFFFF7F7F7")
820 CALL HCHAR(8,15,58)
830 CALL CHAR(59,"8F8FCFCFE7E7E7F3")
840 CALL HCHAR(8,16,59)
850 CALL CHAR(60,"C0FFFFFFFFFFFFFF")
860 CALL HCHAR(8,18,60)
870 CALL CHAR(61,"0F80F8FCFCFEFFFF")
880 CALL HCHAR(8,19,61)
890 CALL CHAR(62,"FF3F037E7F3F3F3F")
900 CALL HCHAR(8,20,62)
910 CALL CHAR(63,"C0F8FE3FF7FFFFFF")
920 CALL HCHAR(8,21,63)
930 CALL CHAR(64,"0000000080C0C0E0")
940 CALL HCHAR(8,22,64)
950 CALL CHAR(65,"0000000003070F3F")
960 CALL HCHAR(9,8,65)
970 CALL CHAR(66,"0F3E7DF3EFDFFFFD")
980 CALL HCHAR(9,9,66)
990 CALL CHAR(67,"9E7DFFFFFCE0C3FF")
1000 CALL HCHAR(9,10,67)
1010 CALL CHAR(68,"7FF8E000001FFFFF")
1020 CALL HCHAR(9,11,68)
1030 CALL CHAR(69,"800000037FFFFFFF")
1040 CALL HCHAR(9,12,69)
1050 CALL CHAR(70,"0306060DFFFFFFFF")
1060 CALL HCHAR(9,13,70)
1070 CALL CHAR(71,"6F6FFFFFFFFF8F8F")
1080 CALL HCHAR(9,14,71)
1090 CALL CHAR(72,"F7FFFFFFFFFF0703")
1100 CALL HCHAR(9,15,72)
1110 CALL CHAR(73,"F3F3FFFFFFFF80C0")
1120 CALL HCHAR(9,16,73)
1130 CALL CHAR(74,"FFFFFFFFFFFF0000")
1140 CALL HCHAR(9,17,74,2)
1150 CALL CHAR(75,"FFFFFFFFFFFF3F00")
1160 CALL HCHAR(9,19,75)
1170 CALL CHAR(76,"9FDFCFE7FFFFFF0F")
1180 CALL HCHAR(9,20,76)
1190 CALL CHAR(77,"F0F0F8FCFCFEFFFF")
1200 CALL HCHAR(9,22,77)
1210 CALL CHAR(78,"000103070F0F0703")
1220 CALL HCHAR(10,7,78)
1230 CALL CHAR(79,"E0F0783C1E0F0703")
1240 CALL HCHAR(10,16,79)
1250 CALL CHAR(80,"00000000000080C0")
1260 CALL HCHAR(10,17,80)
1270 CALL CHAR(81,"FF0F030000000000")
1280 CALL HCHAR(10,22,81)
1290 CALL CHAR(82,"80C0E0F0783C1E0F")
1300 CALL HCHAR(10,23,82)
1310 CALL CHAR(83,"FFFF7F1F0F070301")
1320 CALL HCHAR(11,8,83)
1330 CALL CHAR(84,"0100000000000000")
1340 CALL HCHAR(10,15,84)
1350 CALL CHAR(85,"FF7FCFF6FEFFFFFF")
1360 CALL HCHAR(11,7,84)
1370 CALL HCHAR(11,9,85)
1380 CALL CHAR(86,"FFFFFFFF0FC0F0F8")
1390 CALL HCHAR(11,10,86)
1400 CALL CHAR(87,"FFFFFFFFFFFF0700")
1410 CALL HCHAR(11,11,87)
1420 CALL CHAR(88,"FFFFFEFEFEFEFEFE")
1430 CALL HCHAR(11,13,88)
1440 CALL HCHAR(11,16,84)
1450 CALL CHAR(89,"E0F0783C1C0E0F07")
1460 CALL HCHAR(11,17,89)
1470 CALL CHAR(90,"0000000000018FFF")
1480 CALL HCHAR(11,18,90)
1490 CALL CHAR(91,"000000013FFFF000")
1500 CALL HCHAR(11,19,91)
1510 CALL CHAR(92,"000000FFFFF00000")
1520 CALL HCHAR(11,20,92)
1530 CALL CHAR(93,"000000FFFF070000")
1540 CALL HCHAR(11,21,93)
1550 CALL CHAR(94,"000000C0FEFF0700")
1560 CALL HCHAR(11,22,94)
1570 CALL CHAR(95,"0703010000E0FE7F")
1580 CALL HCHAR(11,23,95)
1590 CALL CHAR(96,"80C0E0F0783C1ECF")
1600 CALL HCHAR(11,24,96)
1610 CALL CHAR(97,"FF7F3F1F07030100")
1620 CALL HCHAR(12,9,97)
1630 CALL CHAR(98,"0080E0F8FEFFFFFF")
1640 CALL HCHAR(12,11,98)
1650 CALL CHAR(99,"1F0700000080E0F8")
1660 CALL HCHAR(12,12,99)
1670 CALL CHAR(100,"FEFEFF7F6C6C6C7C")
1680 CALL HCHAR(12,13,100)
1690 CALL CHAR(101,"0000E0FCFF0F0100")
1700 CALL HCHAR(12,14,101)
1710 CALL CHAR(102,"0000000000E0FC3F")
1720 CALL HCHAR(12,15,102)
1730 CALL CHAR(103,"0301000103030707")
1740 CALL HCHAR(12,17,103)
1750 CALL CHAR(104,"F8E0F0FBFFBF1F0F")
1760 CALL HCHAR(12,18,104)
1770 CALL CHAR(105,"000000FFF7F7FF3")
1780 CALL HCHAR(12,19,105)
1790 CALL CHAR(106,"000000FFFFFF0F9F")
1800 CALL HCHAR(12,20,106)
1810 CALL CHAR(107,"000000E0FFFFFFE1")
1820 CALL HCHAR(12,21,107)
1830 CALL HCHAR(12,22,40)
```

```
1840 CALL HCHAR(12,23,40)
1850 CALL CHAR(110,"FFFF0
   F030000F8FF")
1860 CALL HCHAR(12,24,110
   )
1870 CALL CHAR(111,"8080E
   0E0F07038FC")
1880 CALL HCHAR(12,25,111
   )
1890 CALL CHAR(112,"7F3F1
   F0703010000")
1900 CALL HCHAR(13,10,112
   )
1910 CALL CHAR(113,"FFFFF
   FFFFFFDFE3F")
1920 CALL HCHAR(13,11,113
   )
1930 CALL CHAR(114,"FFFFF
   FFFFFFFDF0F")
1940 CALL HCHAR(13,12,114
   )
1950 CALL CHAR(115,"1CC0F
   0F8FFFFFFFF")
1960 CALL HCHAR(13,13,115
   )
1970 CALL CHAR(116,"00000
   00000C0F0FC")
1980 CALL HCHAR(13,14,116
   )
1990 CALL CHAR(117,"07010
   00000000000")
2000 CALL HCHAR(13,15,117
   )
2010 CALL CHAR(118,"E0F87
   E1F07010000")
2020 CALL HCHAR(13,16,118
   )
2030 CALL CHAR(119,"070E0
   E8EEEFE7F3F")
2040 CALL HCHAR(13,17,119
   )
2050 CALL CHAR(120,"07030
   100000000C0")
2060 CALL HCHAR(13,18,120
   )
2070 CALL CHAR(121,"E3F7F
   FFF7F3F1C06")
2080 CALL HCHAR(13,19,121
   )
2090 CALL CHAR(122,"9E9C1
   C0FFFFF0000")
2100 CALL HCHAR(13,20,122
   )
2110 CALL CHAR(123,"E060E
   0C0F0FF7F00")
2120 CALL HCHAR(13,21,123
   )
2130 CALL CHAR(124,"FF7F3
   F1F0FFFFF01")
2140 CALL HCHAR(13,22,124
   )
2150 CALL CHAR(125,"FFFD9
   FBFFFF7E3FF")
2160 CALL HCHAR(13,24,125
   )
2170 CALL CHAR(126,"FCFEF
```



```
   F3F77F7E3FF")
2180 CALL HCHAR(13,25,126
   )
2190 CALL CHAR(127,"00000
   000808080C0")
2200 CALL HCHAR(13,26,127
   )
2210 CALL CHAR(128,"1F0F0
   70300000000")
2220 CALL HCHAR(14,11,128
   )
2230 CALL CHAR(129,"C3E1F
   1F9FD7F3F0F")
2240 CALL HCHAR(14,12,129
   )
2250 CALL CHAR(130,"00C0F
   0FC9FC7E1F0")
2260 CALL HCHAR(14,15,130
   )
2270 CALL CHAR(131,"00000
   00080E0FC3F")
2280 CALL HCHAR(14,16,131
   )
2290 CALL CHAR(132,"0F0F0
   C0C0C0C0C8C")
2300 CALL HCHAR(14,17,132
   )
2310 CALL CHAR(133,"F0FCF
   E3F07010000")
2320 CALL HCHAR(14,18,133
   )
2330 CALL CHAR(134,"07070
   F9FFCFE7F3F")
2340 CALL HCHAR(14,19,134
   )
2350 CALL CHAR(135,"FFFFE
   0F0FC3FFFFF")
2360 CALL HCHAR(14,20,135
   )
2370 CALL CHAR(136,"F8FF1
   F000000FCFF")
2380 CALL HCHAR(14,21,136
   )
2390 CALL CHAR(137,"00FCF
   F07000000FF")
2400 CALL HCHAR(14,22,137
   )
2410 CALL CHAR(138,"7F00E
   0FF070303FF")
2420 CALL HCHAR(14,23,138
   )
2430 CALL CHAR(139,"C0001
   FFFFF0080C0")
2440 CALL HCHAR(14,24,139
   )
2450 CALL CHAR(140,"7F03F
   0FFFF1F0000")
2460 CALL HCHAR(14,25,140
   )
2470 CALL CHAR(141,"C0E0E
   0F0F0F0F078")
2480 CALL HCHAR(14,26,141
   )
2490 CALL CHAR(142,"07030
   10000000000")
2500 CALL HCHAR(15,12,142
   )
2510 CALL CHAR(143,"FFFFF
   FFF7F1F0F07")
2520 CALL HCHAR(15,13,143
   )
2530 CALL CHAR(144,"F0F8F
   EFFFFFFFFFF")
2540 CALL HCHAR(15,15,144
   )
2550 CALL CHAR(145,"07010
   0C0F0FE3F8F")
2560 CALL HCHAR(15,16,145
   )
2570 CALL CHAR(146,"FCFF3
   F0F0E0ECEFE")
2580 CALL HCHAR(15,17,146
```

```
)
2590 CALL CHAR(147,"0080F
   0FF7F070000")
2600 CALL HCHAR(15,18,147
   )
2610 CALL CHAR(148,"30707
   0F0E0E06060")
2620 CALL HCHAR(15,19,148
   )
2630 CALL CHAR(149,"7F070
   7070F0F0F0F")
2640 CALL HCHAR(15,20,149
   )
2650 CALL CHAR(150,"00FFF
   FFFFFFFFFFF")
2660 CALL HCHAR(15,25,150
   )
2670 CALL CHAR(151,"78F8F
   8F0F0E0E0E0")
2680 CALL HCHAR(15,26,151
   )
2690 CALL HCHAR(16,13,117
   )
2700 CALL CHAR(152,"FFFF7
   F3F1F0F0703")
2710 CALL HCHAR(16,14,152
   )
2720 CALL CHAR(153,"81C0C
   0E0F0F8FFFF")
2730 CALL HCHAR(16,16,153
   )
2740 CALL CHAR(154,"FE3F0
   F07010000F0")
2750 CALL HCHAR(16,17,154
   )
2760 CALL CHAR(155,"00C0F
   8FFCFC30000")
2770 CALL HCHAR(16,18,155
   )
2780 CALL CHAR(156,"60606
   0F0FEFFFF3F")
2790 CALL HCHAR(16,19,156
   )
2800 CALL CHAR(157,"1F1F1
   F0F0FFFFFFF")
2810 CALL HCHAR(16,20,157
   )
2820 CALL HCHAR(17,15,97)
2830 CALL HCHAR(17,24,74)
2840 CALL HCHAR(18,16,97)
2850 CALL HCHAR(17,20,159
   ,3)
2860 CALL HCHAR(19,17,112
   )
2870 CALL HCHAR(19,18,159
   )
2880 CALL HCHAR(19,20,74,
   3)
2890 CALL HCHAR(19,24,116
   )
2900 CALL HCHAR(20,18,117
   )
2910 CALL HCHAR(20,19,128
   )
2920 CALL HCHAR(20,23,117
   )
2930 CALL CHAR(108,"FEFEF
   CF8F0800000")
2940 CALL HCHAR(17,25,108
   )
2950 CALL CHAR(109,"FCFCF
   CFCFCFCFCF8")
2960 CALL HCHAR(18,23,109
   )
2970 CALL HCHAR(19,19,159
   )
2980 CALL HCHAR(19,23,159
   )
2990 GOTO 2990
```

Fenêtrage

Jean-Philippe Guillemant

Sur des micro-ordinateurs bien plus puissants et bien plus modernes que notre bon vieux TI-99/4A, la grande mode est, aujourd'hui, de travailler en mode 'multi-fenêtres'.

Avec le programme ci-après, vous vous apercevrez que nous n'avons rien à envier à d'autres.

Le mode 'multi-fenêtres' fonctionne aussi bien en Basic simple qu'en Basic étendu, à condition de disposer du module assembleur.

Les fichiers

SCROLL_SA

C'est le source du programme assembleur qui permet, à partir d'un programme en Basic simple ou en Basic étendu, la gestion des fenêtres d'écran. Le programme est identique pour les deux versions de Basic, seule une ligne change :

- en Basic simple, il faut mettre, en début de programme (ligne 37), la directive

 `COPY "DSK1.BSCSUP SB"`

 et créer le fichier objet sous le nom

 `"SCROLL_AB"`

- en Basic étendu, il faut remplacer cette ligne par la directive

 `COPY "DSK1.SCROLL SAX"`

 et créer le fichier objet sous le nom

 `"SCROLL AX"`

Bien entendu, on ne peut pas alors assembler le programme en code compressé.

BSCSUP_SB

C'est le désassemblage du programme BSCSUP de la disquette assembleur qui contient les utilitaires de passage d'arguments entre le Basic et l'Assembleur. Nous avons trouvé plus pratique d'avoir un seul programme à charger à l'exécution. Nous avons conservé l'ensemble du programme, mais seules les routines "NUMREF", "STRREF" et "ERR" (cf. paragr. 17.2 du manuel Assembleur) sont ici utiles.

SCROLL_SAX

Il contient les quelques équivalences nécessaires au Basic étendu, puisque les utilitaires sont inclus dans le module.

SCROLL_B

C'est un programme, en Basic simple, qui montre les possibilités de gestion de fenêtres. Il peut gérer, en répondant au menu, soit une fenêtre paramétrable par l'utilisateur, soit cinq fenêtres pré-paramétrées. Ces deux exemples montrent toute la richesse du mode 'multi-fenêtres' et leur but est de donner des idées de programmes.

SCROLL_X

C'est le même programme que ci-dessus, mais en Basic étendu. Les fonctionnalités en sont identiques, seule la programmation diffère.

Utilisation

Lorsque le programme est assemblé, et chargé en mémoire par :

```
CALL INIT
CALL LOAD("DSK1.SCROLL AB")
```

en Basic simple, ou par :

```
CALL INIT
CALL LOAD("DSK1.SCROLL AX")
```

en Basic étendu, on a accès à deux fonctions supplémentaires.

WINDOW

"WINDOW" permet de définir les fenêtres. Elle est appelée par :

```
CALL LINK("WINDOW",
  NF,LH,LB,CG,CD)
```

où les paramètres sont :

NF : numéro de fenêtre ; peut varier de 1 à 5, ce qui permet de définir jusqu'à 5 fenêtres se chevauchant éventuellement. Ce paramètre est facultatif ; s'il est omis, cela signifie qu'on désire définir une seule fenêtre (le cas le plus fréquent) ou bien la fenêtre numéro 1.

LH : ligne haute de la fenêtre (de 1 à 24).

LB : ligne basse de la fenêtre (de 1 à 24).

CG : colonne gauche de la fenêtre (de 1 à 32).

CD : colonne droite de la fenêtre (de 1 à 32). Toutes les colonnes de l'écran peuvent être utilisées, comme avec les instructions CALL HCHAR et CALL VCHAR. Ceci implique que, pour redéfinir la fenêtre Basic normale, il faut mettre comme paramètres 1,24,3,30.

En cas d'erreur, la fonction affiche le message "BAD ARGUMENT" (ou, éventuellement, en Basic étendu, le message "INCORRECT ARGUMENT LIST") et redéfinit la fenêtre par défaut (1,24,1,32).

SCROLL

SCROLL permet l'utilisation des fenêtres ainsi définies. Elle est appelée par

CALL LINK("SCROLL",NF,CH$)

avec, comme paramètres :

NF : est le même numéro de fenêtre que dans la fonction "WINDOW", et est également facultatif (la valeur par défaut est 1).

CH$: est la chaîne à afficher. Il n'y a pas de restrictions sur la longueur de cette chaîne (de 0 à 255 caractères). Si elle est trop longue, seuls les premiers caractères sont pris en compte. Si elle est trop courte, elle est remplie de blancs.

Bien entendu, tous les paramètres peuvent être soit des valeurs immédiates, soit des variables.

Programme "Scroll_B"

Programme de démonstration, version 'Basic TI'

```
100 CALL CLEAR
110 PRINT "chargement cod
   e assembleur"
120 CALL INIT
130 CALL LOAD("DSK1.SCROL
   L AB")
140 CALL CLEAR
150 PRINT "VOULEZ-VOUS UN
    EXEMPLE":"        - D'U
   NE FENETRE (1)":"
    - DE 5 FENETRES (2)"
160 INPUT "OU VOULEZ-VOUS
    SORTIR (0):":I
170 IF I>2 THEN 140
180 ON I+1 GOTO 220,270,5
   40
190 REM
200 REM  sortie
210 REM
220 CALL CLEAR
230 STOP
240 REM
250 REM   1 fenetre param
   etree
260 REM
270 PRINT :"ENTREZ LA DEF
   INITION DE LA  FENETRE
   :":
280 INPUT "LIGNE HAUTE (1
   -24): ":LH
290 INPUT "LIGNE BASSE (1
   -24): ":LB
300 INPUT "COLONNE GAUCHE
    (1-32): ":CG
310 INPUT "COLONNE DROITE
    (1-32): ":CD
320 REM  remplissage de l
   'ecran
330 A$="12345678901234567
   8901234567890123456789
   0"
340 CALL LINK("WINDOW",1,
   24,1,32)
350 FOR I=1 TO 24
360 CALL LINK("SCROLL",A$
   )
370 A$=SEG$(A$,2,39)&SEG$
   (A$,1,1)
380 NEXT I
390 CALL LINK("WINDOW",23
   ,24,5,28)
400 CALL LINK("SCROLL","
    TAPEZ SUR UNE TOUCHE"
   )
410 CALL LINK("SCROLL","
     ENTER POUR FINIR")
420 REM  fenetre utilisat
   eur
430 CALL LINK("WINDOW",LH
   ,LB,CG,CD)
440 I=0
450 CALL KEY(0,K,S)
460 IF S=0 THEN 450
470 IF K=13 THEN 140
480 CALL LINK("SCROLL","
   FENETRE "&STR$(I))
490 I=I+1
500 GOTO 450
510 REM
520 REM   5 fenetres
530 REM
540 CALL CLEAR
```

```
550 CALL LINK("WINDOW",1,
   11,1,16)
560 CALL LINK("WINDOW",2,
   1,11,17,32)
570 CALL LINK("WINDOW",3,
   12,22,1,16)
580 CALL LINK("WINDOW",4,
   12,22,17,32)
590 CALL LINK("WINDOW",5,
   6,17,8,23)
600 PRINT " TAPEZ SUR UNE
    DES TOUCHES":"  1 A 5
   , ENTER POUR FINIR";
610 CALL KEY(0,K,S)
620 IF S=0 THEN 610
630 IF K=13 THEN 140
640 IF (K<49)*(K>53)THEN
   610
650 CALL LINK("SCROLL",K-
   48,"**"&CHR$(K)&" FENE
   TRE "&CHR$(K)&"**")
660 GOTO 610
```

Programme "Scroll_X"

Programme de démonstration, version 'Basic Étendu'

```
100 DISPLAY AT(24,1)ERASE
    ALL:"chargement code
   assembleur" :: CALL IN
   IT :: CALL LOAD("DSK1.
   SCROLL AX")
110 DISPLAY AT(1,1)ERASE
   ALL:"VOULEZ-VOUS UN EX
   EMPLE":"        - D'UNE
   FENETRE (1)":"       -
   DE 5 FENETRES (2)":"OU
    VOULEZ-VOUS SORTIR (0
   ):"
120 ACCEPT AT(4,28)SIZE(1
   )VALIDATE(DIGIT)BEEP:I
    :: IF I>2 THEN 120 EL
   SE ON I+1 GOTO 160,200
   ,340
130 !
140 !sortie
150 !
160 CALL CLEAR :: STOP
170 !
180 ! 1 fenetre parametra
   ble
190 !
200 DISPLAY AT(6,1)BEEP:"
   ENTREZ LA DEFINITION D
   E LA  FENETRE:": :"LIG
   NE HAUTE (1-24):":"LIG
   NE BASSE (1-24):"
210 DISPLAY AT(11,1):"COL
   ONNE GAUCHE (1-32):":"
   COLONNE DROITE (1-32):
   "
220 ACCEPT AT(9,24):LH ::
    ACCEPT AT(10,24):LB :
   : ACCEPT AT(11,24):CG
   :: ACCEPT AT(12,24):CD
230 ! remplissage de l'ec
   ran
240 CALL LINK("WINDOW",1,
   24,1,32):: A$="1234567
   8901234567890123456789
   01234567890"
250 FOR I=1 TO 24 :: CALL
    LINK("SCROLL",A$):: A
   $=SEG$(A$,2,39)&SEG$(A
   $,1,1):: NEXT I
260 DISPLAY AT(23,4)SIZE(
   22):" TAPEZ SUR UNE TO
   UCHE" :: DISPLAY AT(24
   ,4)SIZE(22):"   ENTER
   POUR FINIR";
270 ! fenetre utilisateur
280 CALL LINK("WINDOW",LH
   ,LB,CG,CD):: I=0
290 CALL KEY(0,K,S):: IF
   NOT S THEN 290 ELSE IF
    K=13 THEN 110
300 CALL LINK("SCROLL","F
   ENETRE "&STR$(I)):: I=
   I+1 :: GOTO 290
310 !
320 ! 5 fenetres
330 !
340 CALL CLEAR :: CALL LI
   NK("WINDOW",1,11,1,16)
350 CALL LINK("WINDOW",2,
   1,11,17,32)
360 CALL LINK("WINDOW",3,
   12,22,1,16)
370 CALL LINK("WINDOW",4,
   12,22,17,32)
380 CALL LINK("WINDOW",5,
   6,17,8,23)
390 DISPLAY AT(23,1):" TA
   PEZ SUR UNE DES TOUCHE
   S":"  1 A 5, ENTER POU
   R FINIR";
400 CALL KEY(0,K,S):: IF
   NOT S THEN 400 ELSE IF
    K=13 THEN 110
410 IF (K>48)AND(K<54)THE
   N CALL LINK("SCROLL",K
   -48,"**"&CHR$(K)&" FEN
   ETRE "&CHR$(K)&"**")
420 GOTO 400
```

Source 'SCROLL_SA'

```
      TITL 'SCROLL - EQUIVALENCES & RESERVATIONS'
*********************************************
*                                           *
*                                           *
*   SSSS   CCCC   RRR    OOO   L      L     *
*  S       C      R   R  O   O L      L     *
*  S       C      R   R  O   O L      L     *
*   SSS    C      RRRR   O   O L      L     *
*      S   C      R R    O   O L      L     *
*      S   C      R  R   O   O L      L     *
*  SSSS    CCCC   R   R   OOO  LLLLL  LLLLL *
*                                           *
*                                           *
* * * * * * Jean-Philippe  GUILLEMANT * * * * * *
*                                           *
*********************************************


      DEF  WINDOW      definition de la fenetre
*                      parametres: no fenetre (facultatif)
*                                  ligne haute
*                                  ligne basse
*                                  colonne gauche
*                                  colonne droite
      DEF  SCROLL      routine de rouleau
*                      parametres: no fenetre (facultatif)
*                          chaine a inserer (peut etre vide)

***********************************
* Selon la version utilisee (respecti- *
* vement BASIC simple ou BASIC etendu),*
* il faut inserer ici une directive    *
* COPY, respectivement de:             *
*                   DSK1.BSCSUP_SB     *
*             ou de DSK1.SCROLL_SAX    *
***********************************

       COPY "DSK1.SCROLL_SAX" ou bien "DSK1.BSCSUP_SB"

* reservations memoire

BUFFS  BSS  32           buffer ligne d'ecran
BUFFA  BSS  256          buffer derniere ligne (param SCROLL)

LINH   DATA 0,0,0,0,0       lere ligne chaque fenetre (0 << 23)
LINB   DATA 23,23,23,23,23 dern ligne chaque fenetre (0 << 23)
COLG   DATA 0,0,0,0,0       col gauche chaque fenetre (0 << 31)
COLD   DATA 31,31,31,31,31 col droite chaque fenetre (0 << 31)

UN     BYTE 1
DEUX   BYTE 2
QUATRE BYTE 4
CINQ   BYTE 5
BLANC  BYTE ' '
       EVEN
       TITL 'SCROLL - INITIALISATION (WINDOW)'
       PAGE
```

```
* routine d'initialisation

WINDOW EQU  $
       MOV  R11,R10        sauver adr. de retour
       CLR  R9             1ere fenetre par defaut
       LI   R8,1           no d'argument
       CB   @NBRARG,@QUATRE 4 arguments ?
       JEQ  WIND1          br si oui, 1ere fenetre
       CB   @NBRARG,@CINQ 5 arguments ?
       JNE  WERRAL         br si non
       BL   @GETARG        no de fenetre
       DATA 5-1            1 < < 5 ?
       JMP  WERRBA         si erreur
       MOV  R2,R9          index parametres de la fenetre
       SLA  R9,1           en mots
       INC  R8             argument suivant
WIND1  BL   @GETARG        ligne haute
LIGMAX DATA 24-1           1 < < 24 ?
       JMP  WERR           si erreur
       MOV  R2,@LINH(R9)
       INC  R8             argument suivant
       BL   @GETARG        ligne basse
       DATA 24-1           1 < < 24 ?
       JMP  WERR           si erreur
       C    R2,@LINH(R9) lignes coherentes ?
       JL   WERR           br si non
       MOV  R2,@LINB(R9)
       INC  R8             argument suivant
       BL   @GETARG        colonne gauche
COLMAX DATA 32-1           1 < < 32 ?
       JMP  WERR           si erreur
       MOV  R2,@COLG(R9)
       INC  R8             argument suivant
       BL   @GETARG        colonne droite
       DATA 32-1           1 < < 32 ?
       JMP  WERR           si erreur
       C    R2,@COLG(R9) colonnes coherentes ?
       JL   WERR           br si non
       MOV  R2,@COLD(R9)
       B    @RETOUR

* erreur de definition de fenetre
*    remettre les valeurs par defaut (fenetre = ecran total)

WERR   EQU  $
       CLR  @LINH(R9)             ligne haute
       MOV  @LIGMAX,@LINB(R9) ligne basse
       CLR  @COLG(R9)             colonne gauche
       MOV  @COLMAX,@COLD(R9) colonne droite
WERRBA LI   R0,ERRBA       message 'bad argument'
       JMP  $+4
WERRAL LI   R0,ERRIAL      nessage 'incorrect argument list'
       BLWP @ERR           affichage erreur
       JMP  RETOUR

* recherche d'un argument numerique
*   entree: R8 = no du parametre
*           argument = valeur max
*   retour: R2 = valeur parametre (-1)

GETARG EQU  $
       CLR  R0
       MOV  R8,R1
       BLWP @NUMREF        argument dans FAC
       BLWP @XMLLNK        link to ROM routines
       DATA CFI            convert to integer
       MOV  @FAC,R2
       DEC  R2             -1 (offset 0)
       JLT  GETAR1         br si <= 0
       C    R2,*R11
       JGT  GETAR1         br si trop grand
       INCT R11            si correct
GETAR1 INCT R11            si erreur
       RT

       TITL 'SCROLL - EXECUTION (SCROLL)'
       PAGE

* execution du scroll

SCROLL EQU  $
       MOV  R11,R10        sauver adresse de retour
       CLR  R9             1ere fenetre par defaut
       LI   R8,1           no d'argument
       CB   @NBRARG,@UN    1 argument ?
       JEQ  SCROL1         br si oui, 1ere fenetre
       CB   @NBRARG,@DEUX 2 arguments ?
       JNE  WERRAL         br si non
       BL   @GETARG        no de fenetre
       DATA 5-1            <= 5
       JMP  WERRBA         si erreur
       MOV  R2,R9          index parametres de la fenetre
       SLA  R9,1           en mots
       INC  R8             argument suivant

SCROL1 LI   R2,BUFFA       adresse du buffer
       SETO R1             longueur max de chaine (= 255)
       MOVB R1,*R2
       CLR  R0
       MOV  R8,R1          numero d'argument
       BLWP @STRREF        recherche de la ligne a ecrire

       MOV  @COLD(R9),R3
       S    @COLG(R9),R3
       INC  R3             nbre de caracteres a scroller
       MOV  R3,R4
       SWPB R4
       CB   *R2,R4         chaine assez grande ?
       JHE  SCROL3         oui
       SB   *R2,R4         non, la remplir de blancs
       SWPB R4
       CLR  R1
       MOVB *R2+,R1
       SWPB R1
       A    R1,R2          adresse fin de chaine
SCROL2 MOVB @BLANC,*R2+
       DEC  R4
       JNE  SCROL2

SCROL3 MOV  @LINB(R9),R6
       MOV  @LINH(R9),R5 ligne superieure
       S    R5,R6          nombre de lignes a scroller
       SLA  R5,5           * 32 (adresse 1ere ligne a scroller)
       A    @COLG(R9),R5 + 1ere colonne
       MOV  R5,R4

SCROL4 DEC  R6             fini ?
       JLT  SCROL7         br si oui, traiter derniere ligne
       AI   R4,32          adresse ligne suivante
       MOV  R4,R0
       BL   @VDPADR        selection adresse VDP
       LI   R1,BUFFS
       MOV  R3,R2          longueur a lire
SCROL5 MOVB @VDPRD,*R1+    lire une ligne
       DEC  R2
       JNE  SCROL5

       MOV  R5,R0
       BL   @VDPADW        selection adresse VDP
       LI   R1,BUFFS
       MOV  R3,R2          longueur a ecrire
SCROL6 MOVB *R1+,@VDPWD    ecrire une ligne
       DEC  R2
       JNE  SCROL6
       AI   R5,32          pour ligne suivante
       JMP  SCROL4         boucle

SCROL7 MOV  R5,R0          derniere ligne
       BL   @VDPADW        selection adresse VDP
       LI   R1,BUFFA+1
       MOV  R3,R2          longueur a ecrire
SCROL8 MOVB *R1+,R0
       AI   R0,>6000       offset BASIC
       MOVB R0,@VDPWD
       DEC  R2
       JNE  SCROL8


* retour au BASIC

RETOUR EQU  $
       CLR  R0
       MOVB R0,@STATUS     pas d'erreurs
       B    *R10           retour

* selection adresse VDP *
```

```
VDPADW EQU  $              pour ecriture
       ORI  R0,>4000
VDPADR EQU  $              pour lecture
       SWPB R0
       MOVB R0,@VDPWA
       SWPB R0
       MOVB R0,@VDPWA
       RT

       END
```

Source 'SCROLL_SAX'

```
****************************************
*                                      *
* S C R O L L - B A S I C  e t e n d u *
*                                      *
* Ce fichier doit etre inclus dans le  *
* programme SCROLL_SA pour l'utilisa-  *
* tion en BASIC etendu.                *
*
NUMREF EQU >200C        routine 'get numeric parameter'
STRREF EQU >2014        routine 'get string parameter'
XMLLNK EQU >2018        ROM routine access
ERR    EQU >2034        error reporting routine
NBRARG EQU >8312        nombre d'arguments
FAC    EQU >834A        floating point acc.
STATUS EQU >837C        GPL status return
VDPRD  EQU >8800        VDP RAM read data
VDPWD  EQU >8C00        VDP RAM write data
VDPWA  EQU >8C02        VDP RAM write address
*
CFI    EQU >12B8        routine 'convert float. to int.'
ERRBA  EQU >1C00        erreur 'bad argument
ERRIAL EQU >1F00        erreur 'incorrect argument list'
*
****************************************
```

Source 'BSCSUP_SB'

```
*************************
*                       *
*    UTILITAIRES        *
*                       *
*    BASIC SIMPLE       *
*                       *
*  desassemblage de     *
*                       *
*    DSK1.BSCSUP        *
*                       *
*  J.-Ph. GUILLEMANT    *
*                       *
*************************

*      DEF  NUMASG      numeric assignment
*      DEF  STRASG      string assignment
*      DEF  NUMREF      get numeric parameter
*      DEF  STRREF      get string parameter
*      DEF  ERR         error reporting

       REF  XMLLNK      link to ROM routine
       REF  GPLWS       GPL workspace
       REF  GPLLNK      link to GPL routines
       REF  UTLTAB      utility variable table

CFI    EQU  >1200       routine 'convert float. to int'

BASERR EQU  >00CE       routine d'erreur BASIC
*      EQU  >1EAA       (cf. ALLCHN)
*      EQU  >1788       (cf. ALLCHN)
IDLIST EQU  >200A       identifier list
STRLNG EQU  >830C       longueur chaine
ADRARG EQU  >8310       adr. des descripteurs d'arguments
NBRARG EQU  >8312       nombre d'arguments du CALL LINK
STRSPC EQU  >831C       allocated string space
NOERR  EQU  >8322       numero d'erreur
OPTBAS EQU  >8343       option base
FAC    EQU  >834A       floating point accumulator
*      EQU  >8350       (cf. ALLCHN)
STATUS EQU  >837C       GPL status return
VDPRD  EQU  >8800       VDP read data reg.
VDPWD  EQU  >8C00       VDP write data reg.
VDPWA  EQU  >8C02       VDP write addresse reg.

* codes d'erreurs
ERRSNM EQU  >1500       erreur 'string-numbre mismatch'
ERRBA  EQU  >1600       erreur 'bad argument'
ERRIAL EQU  >1600       erreur 'incorrect argument list'
ERRRBS EQU  >1700       erreur 'bad subsript'
ERRINC EQU  >2500       erreur inconnue

NUMASG DATA UTLWS,ASGNUM numeric assignment
NUMREF DATA UTLWS,REFNUM get numeric parameter
STRASG DATA UTLWS,ASGSTR string assignment
STRREF DATA UTLWS,REFSTR get string parameter
ERR    DATA UTLWS,UERR

UTLWS  BSS  32          working storage area

STRTAG BYTE >65         identifieur de chaine
       EVEN

*************************
* NUMASG: numeric assignment to BASIC
*   R0 = 0 si variable numerique
*        ou numero d'element si tableau
*   R1 = numero d'argument de LINK
*************************

ASGNUM MOV  *R13,R0       user R0
       MOV  @2(R13),R1    user R1
       BL   @PARARG       parametres de l'argument
       MOVB *R3,R3        est-ce une constante num. ?
       JEQ  BADERR        oui, 'bad argument'
       SRL  R3,8
       DECT R3            variable ou element tableau num. ?
       JNE  ASGNM3        non, ce doit etre un tableau num.
       MOV  R0,R0         oui, a t-on demande une variable ?
       JNE  BADERR        non, 'bad argument'
       MOV  R5,R3         adresse du descripteur
       INCT R3            + 2
       BL   @VDP1RD       lire l'octet 2
       JNE  SNMERR        qui doit etre nul !
       INCT R3            + 4
       BL   @VDP2RD       lire l'adresse de la variable
       MOV  R1,R3
ASGNM1 MOV  R3,R4         adresse de la variable en VDP RAM
       LI   R3,FAC        adresse de la valeur a transferer
       LI   R2,4
ASGNM2 MOV  *R3,R1
       BL   @VDP2WT       ecriture de la variable (4*2 octets)
       INCT R3
       INCT R4
       DEC  R2            fini ?
       JGT  ASGNM2        non, boucle
       RTWP               retour vers l'utilisateur

ASGNM3 BL   @TABNUM       traitement tableau
       JMP  ASGNM1
* recherche des parametres d'un argument
*   entree: R1 = numero d'argument
*   retour: R3 = pointeur sur l'identifieur
*           R5 = adresse du descripteur en VDP RAM
PARARG MOV  R1,R1         numero d'argument
       JEQ  BADERR        si nul, c'est pas bon
       SLA  R1,8
       CB   @NBRARG,R1    > que le nombre reel ?
       JLT  BADERR        oui, c'est pas bon
       SRL  R1,8
       MOV  R1,R5
       SLA  R5,3           * 8
       AI   R5,8           + 8
       A    @ADRARG,R5
       MOV  R1,R3         numero de l'argument
       DEC  R3            offset 0
       AI   R3,IDLIST     adresse identifieur dans la liste
       RT
* traitement tableau numerique
*   entree: R3 = identifier - 2
*           R5 = adr. descripteur
```

```
*   retour: R3 = adr. element recherche
TABNUM MOV  R11,R9         sauver adr. de retour
       DECT R3             identifier = 4 ?
       JNE  SNMERR         non, 'string-number mismatch'
       MOV  R5,R3
       BL   @VDP2RD        lire ptr sur array symbol table entry
       MOV  R1,R3
       BL   @VDP1RD        lire array symbol table entry
       JLT  SNMERR         'string-number mismatch'
       BL   @CONTAB
       S    R4,R0
       SLA  R0,3           8 octets par element
       A    R0,R3
       B    *R9            retour
*
*   entree: R0 = numero d'element dans le tableau
*           R1 = 1er octet de l'array symbol table entry
*           R5 = adr. descripteur
*   retour: R3 = adr. 1er element du tableau
*           R4 = option base
CONTAB MOV  R11,R10        sauver adr. de retour
       SLA  R1,5
       SRL  R1,13
       MOV  R1,R8
       MOVB @OPTBAS,R4     option base
       SRL  R4,8
       JEQ  CONTB1         br si option base 0
       DEC  R0             si op. base 1, no d'elem doit etre=0
       JLT  RBSERR         'bad subscript'
       INC  R0
CONTB1 LI   R6,1
       MOV  R5,R3          adr. descripteur
       AI   R3,4           oct 4/5 = ptr sur array's value space
       BL   @VDP2RD        lire ce pointeur
       MOV  R1,R3          adr. des index max.
       DECT R3
CONTB2 INCT R3             index max. de la dimension suivante
       BL   @VDP2RD        lire index max d'une dimension
       INC  R1
       S    R4,R1          offset 0 (= f(option base))
       MPY  R1,R6
       MOV  R6,R6          > 65535 ?
       JNE  RBSERR         oui, 'bad subscript'
       MOV  R7,R6          nbre total d'elements
       DEC  R8             toutes dimensions ?
       JGT  CONTB2         non, boucle
       DEC  R6
       A    R4,R6
       C    R0,R6          no d'elem demande>nbre total d'elem ?
       JGT  RBSERR         oui, 'bad subscript'
       INCT R3             adr. 1er element
       B    *R10           retour

SNMERR LI   R0,ERRSNM      'string-number mismatch'
       B    @UERR1

BADERR LI   R0,ERRBA       'bad argument'
       B    @UERR1

RBSERR LI   R0,ERRRBS      'bad subscript'
       B    @UERR1

**************************
* NUMREF: get numeric parameter from BASIC
*   R0 = 0 si variable
*        ou numero d'element si tableau
*   R1 = numero d'argument de LINK
**************************

REFNUM MOV  *R13,R0        user R0
       MOV  @2(R13),R1     user R1
       BL   @PARARG        recherche parametres
       MOVB *R3,R3         identifier
       SRL  R3,8
       JNE  REFNM2         br si ce n'est pas une expression num
       MOV  R0,R0          si l'argument demande est un tableau,
       JNE  REFNM7         alors, 'bad argument'
* expression numerique
       LI   R2,8
       LI   R4,FAC
       MOV  R5,R3          adr. descript (contient expression)
REFNM1 BL   @VDP2RD        lire la valeur (4 * 2 octets)
       MOV  R1,*R4+        la ranger dans FAC
       INCT R3
       DECT R2             4 * ?
       JGT  REFNM1         boucle si non
       RTWP                retour vers l'utilisateur
*
REFNM2 DECT R3             variable ou element de tableau num. ?
       JNE  REFNM5         br si non
       MOV  R0,R0          type de parametre demande
       JNE  REFNM7         si tableau, 'bad argument'
* variable ou element de tableau
       MOV  R5,R3          adr. descripteur
       INCT R3             octet 2 doit etre nul
       BL   @VDP1RD        le lire
       JNE  REFNM6         si != 0, 'string-number mismatch'
       INCT R3             octets 4 & 5 = ptr sur la variable
       BL   @VDP2RD        lire le pointeur
       MOV  R1,R3
REFNM3 LI   R4,FAC
       LI   R2,4
REFNM4 BL   @VDP2RD        lire la variable (4 mots)
       MOV  R1,*R4+
       INCT R3
       DEC  R2             fini ?
       JGT  REFNM4         boucle si non
       RTWP                retour vers l'utilisateur
* tableau
REFNM5 BL   @TABNUM        controle & recherche element
       JMP  REFNM3

REFNM6 B    @SNMERR        'string-number mismatch'

REFNM7 B    @BADERR        'bad argument'

**************************
* STRASG: string assignment
*   R0 = 0 si variable
*        ou numero d'element si tableau
*   R1 = numero d'argument de LINK
*   R2 = adr. de la chaine
**************************

ASGSTR MOV  *R13,R0        user R0
       MOV  @2(R13),R1     user R1
       MOV  @4(R13),R9     user R2
       BL   @PARARG        recherche parametres
       MOVB *R3,R3         identifieur
       SRL  R3,8
       DEC  R3             = 1 (string expression) ?
       JEQ  ASGST3         br si oui, 'bad argument'
       DECT R3             = 3 (string variable) ?
       JNE  ASGST4         br si non
       MOV  R0,R0          si parametre demande est un tableau,
       JNE  ASGST3         alors, 'bad argument'
* variable chaine
       MOV  R5,R3          adr. descripteur
       INCT R3
       BL   @VDP1RD        lire octet 2
       CB   R1,@STRTAG     il doit etre = >65
       JNE  ASGST2         sinon, 'string-number mismatch'
       LI   R6,8
       LI   R4,FAC
       MOV  R5,R3
ASGST1 BL   @VDP2RD        lire le descripteur dans FAC
       MOV  R1,*R4+
       INCT R3
       DECT R6             8 octets ?
       JGT  ASGST1         boucle si non
       BL   @ALLCHN        ranger la chaine
       AI   R5,4           adr. pointeur chaine dans descripteur
       MOV  R5,R4
       MOV  *R6,R1         adr. chaine en VDP RAM
       BL   @VDP2WT        mettre l'adresse a jour
       INCT R4             adr. longueur chaine dans descripteur
       MOV  @STRLNG,R1
       SRL  R1,8
       BL   @VDP2WT        mettre a jour la longueur
       RTWP                retour vers l'utilisateur

ASGST2 LI   R0,ERRSNM      'string-number mismatch'
       B    @UERR1

ASGST3 LI   R0,ERRBA       'bad argument'
       B    @UERR1
```

```
* traitement tableau chaine
*   entree: R3 = identifieur (-3)
*           R5 = adr. descripteur
*   retour: R1 = adr. element recherche
TABSTR MOV  R11,R2        adr. de retour
       DECT R3            tableau de chaines ?
       JNE  ASGST2        br si non, 'string-number mismatch'
       MOV  R5,R3         adr. du descripteur
       BL   @VDP2RD       lire octets 0 & 1
       MOV  R1,R3         ptr to array symbol table entry
       BL   @VDP1RD       lire 1er octet le la table4
       JLT  TABST1        O.K.
       B    @ASGST2       'string-number mismatch'
TABST1 BL   @CONTAB       control & calc adr. pointeurs d'elem
       S    R4,R0         offset = f(option base)
       SLA  R0,1          en mots
       A    R0,R3         + numero d'element
       BL   @VDP2RD       lire l'adresse de la chaine
       B    *R2           retour
* tableau
ASGST4 BL   @TABSTR       traitement tableau
       LI   R6,FAC        ranger dans FAC nouveau descripteur
       MOV  R3,*R6+
       MOVB @STRTAG,*R6+ indicateur de chaine
       MOVB R4,*R6+       option base (en prime)
       MOV  R1,*R6+       ancienne adresse chaine
       MOV  R1,R3
       JNE  ASGST5        si non nulle
       CLR  *R6           0 comme longueur
       JMP  ASGST6
ASGST5 DEC  R3
       BL   @VDP1RD       lire la longueur de l'ancienne chaine
       SRL  R1,8
       MOV  R1,*R6        la ranger
ASGST6 BL   @ALLCHN       mettre a jour la chaine
       RTWP               retour vers l'utilisateur
* allocation & rangement d'une chaine en VDP RAM
ALLCHN MOV  R11,R2        adr. de retour
       LWPI GPLWS
       LI   R11,>1EAA
       BL   *R11
       LWPI UTLWS
       MOVB *R9,R6        longueur chaine a assigner
       SRL  R6,8
       MOV  R6,@STRLNG
       MOV  R6,@>8350
       BLWP @GPLLNK       link to GPL routines
       DATA >0038         get string space
       LI   R6,FAC
       LI   R4,28
       MOV  R4,*R6+
       MOVB @STRTAG,*R6+
       MOVB R4,*R6+       indicateur de chaine
       MOV  @STRSPC,*R6   adresse allouee
       MOV  @STRLNG,R8    longueur nulle ?
       JEQ  ALLCH2        br si oui
       MOV  *R6,R4
       MOV  R9,R3         adr. chaine
       INC  R3            saut de l'octet de longueur
ALLCH1 MOVB *R3+,R1
       BL   @VDP1WT       copie chaine en VDP RAM
       INC  R4
       DEC  R8            fini ?
       JGT  ALLCH1        boucle si non
ALLCH2 LWPI GPLWS
       LI   R11,>1788
       BL   *R11
       LWPI UTLWS
       MOV  R2,R11        retour
       RT

**************************
* STRREF: get string parameter
*   R0 = 0 si variable ou constante
*        ou numero d'element si tableau
*   R1 = numero d'argument de LINK
*   R2 = adr. ou ranger la chaine
*        1er octet du buffer = longueur max
**************************

REFSTR MOV  *R13,R0       user R0
       MOV  @2(R13),R1    user R1
       BL   @PARARG       recherche parametre
       MOVB *R3,R3        identifieur
       SRL  R3,8
       DEC  R3            = 1 (string expression) ?
       JEQ  REFST1        br si oui
       DECT R3            = 3 (string variable) ?
       JNE  REFST6        br si non, doit etre un tableau
* variable ou constante
REFST1 MOV  R0,R0         user R0 doit etre = 0
       JNE  REFST8        non, 'bad argument'
       MOV  @4(R13),R0    user R2
       MOV  R5,R3         adr. du descripteur
       INCT R3            l'octet 2 doit contenir >65
       BL   @VDP1RD       le lire
       CB   R1,@STRTAG    correct ?  A034
       JNE  REFST7        non, 'string-number mismatch'
       INCT R3
       BL   @VDP2RD       lire octets 4 & 5 (ptr adr. chaine)
REFST2 MOV  R1,R1         adr. chaine nulle ?
       JEQ  REFST3        oui
       MOV  R1,R6
       DEC  R1            octet precedent la chaine = longueur
       MOV  R1,R3
       BL   @VDP1RD       lire la longueur
       CB   *R0,R1        > longueur demandee ?
       JL   REFST9        oui, ????
REFST3 MOVB R1,*R0+       ranger longueur dans 1er octet buffer
       JEQ  REFST5        fini si chaine vide9
       MOV  R6,R3         adr. chaine en VDP RAM
       SRL  R1,8
       MOV  R1,R5         longueur a lire
REFST4 BL   @VDP1RD       lire un octet
       MOVB R1,*R0+       et le ranger dans buffer utilisateur
       INC  R3
       DEC  R5            chaine lue ?
       JGT  REFST4          boucle si non
REFST5 RTWP               retour vers l'utilisateur
* tableau
REFST6 BL   @TABSTR       traitement tableau
       MOV  @4(R13),R0    user R2 (adr. du buffer)
       JMP  REFST2        et copie

REFST7 B    @SNMERR       'string-number mismatch'

REFST8 B    @BADERR       'bad argument'

REFST9 LI   R0,ERRINC     ''
       B    @UERR1
* lecture de 2 octets de la VDP RAM
*   entree: R3 = adr. de lecture
*   retour: R1 = valeur lue
VDP2RD SWPB R3
       MOVB R3,@VDPWA
       SWPB R3
       MOVB R3,@VDPWA
       NOP
       MOVB @VDPRD,R1
       SWPB R1
       MOVB @VDPRD,R1
       SWPB R1
       RT
* ecriture de 2 octets en VDP RAM
*   entree: R1 = mot a ecrire
*           R4 = adr. d'ecriture
VDP2WT SWPB R4
       MOVB R4,@VDPWA
       SWPB R4
       ORI  R4,>4000
       MOVB R4,@VDPWA
       NOP
       MOVB R1,@VDPWD
       SWPB R1
       MOVB R1,@VDPWD
       SWPB R1
       RT
* lecture d'1 octet de la VDP RAM
*   entree: R3 = adr. de lecture
*   retour: R1 = octet lu
VDP1RD SWPB R3
       MOVB R3,@VDPWA
       SWPB R3
       MOVB R3,@VDPWA
       NOP
       MOVB @VDPRD,R1
```

```
       RT
* ecriture d'1 octet en VDP RAM
*   entree: R1 = octet a ecrire
*           R4 = adr. d'ecrirure
VDP1WT SWPB R4
       MOVB R4,@VDPWA
       SWPB R4
       ORI  R4,>4000
       MOVB R4,@VDPWA
       NOP
       MOVB R1,@VDPWD
       RT

*************************
* ERR: error reporting
*   R0 = code d'erreur
*************************
UERR   MOV  *R13,@NOERR
       JMP  UERR2
UERR1  MOV  R0,@NOERR
UERR2  LWPI GPLWS
       LI   R11,UTLTAB     Utility table
       AI   R11,14         GPL return address
       MOV  *R11,R11       -> R11
       B    @BASERR
```

Disques double face

Yann Fasquel

Grande nouvelle pour les utilisateurs du boîtier d'extension du TI-99/4A : il est tout à fait possible d'utiliser des lecteurs de disquettes double face, et de disposer ainsi de 180Ko par unité de disquette.

La carte contrôleur est tout à fait capable de gérer ces lecteurs, non seulement du point de vue 'hard', mais aussi du point de vue 'soft'. Et cela sans aucune modification de la carte contrôleur.

Les problèmes rencontrés jusqu'à présent pour cette mise en œuvre viennent d'une erreur dans les 'softs' Texas : il y a une incompatibilité entre le module "DISK MANAGER" et le 'soft' de la carte contrôleur, car, lors du 'formatage' des disquettes, le programme du module positionne certains octets en fonction des réponses aux questions SIMPLE DENSITE et SIMPLE FACE, mais bas les bons.

Dans l'enchaînement, le 'soft' de la carte contrôleur vient tester ces octets, mais le problème est là : pour la carte contrôleur, l'octet positionné par le module à la question SIMPLE DENSITE correspond au nombre de faces. De la même façon, l'octet positionné par le module à la réponse donnée pour le nombre de faces correspond, pour la carte contrôleur, à la densité. Il suffit donc d'inverser les réponses entre les deux questions pour pouvoir disposer des lecteurs double face si, bien sûr, on dispose d'unités de ce type.

Exemple : répondre aux questions

```
SIMPLE FACE (O/N)? O
SIMPLE DENSITE (O/N)? N
```

et, à ce moment, vous verrez le nombre de secteurs disponibles dépasser le nombre fatidique de 359 et aller jusqu'à 718. Une fois le formatage effectué en double face, le 'soft' du contrôleur reconnaîtra toujours les deux faces, pour tous les accès disque.

Cependant, en ce qui concerne la double densité, il faudrait changer tout le contrôleur, aussi bien la partie 'soft' que la partie 'hard'.

Le Pascal reconnais aussi les deux faces. Ainsi, on dispose de 360 blocs par disquette.

Conseils

- pour le branchement des unités de disquettes, se reporter aux documents fournis par Texas Instruments avec votre contrôleur ;
- n'essayez pas de mettre deux unités demi-hauteur dans le boîtier d'extension, car l'alimentation interne ne tiend pas la puissance demandée, et entre en limitation ;
- sur chaque lecteur, il y a un réseau de résistances qui sert aux adaptations de bus. Il y a deux possibilités de montage :
 - vous mettez vos unités en série sur le réseau et il ne faut laisser le réseau que sur le dernier lecteur ;
 - utilisation du lecteur interne et de lecteurs externes : théoriquement, il faut changer le réseau du lecteur interne par des résistances plus fortes (le double), et sur les lecteurs externes ne laisser le réseau que sur le dernier lecteur sur le bus, réseau qui sera, lui aussi, d'une valeur double. Mais il est possible de ne pas changer ces réseaux. À priori, cela fonctionne correctement ; mon système est monté ainsi ;
- ne pas oublier de programmer le numéro de disque sur chaque lecteur, et ceci en fonction de chaque type de lecteur ;
- on peut prendre n'importe quel type de lecteur du moment qu'il est au standard SHUGART, notamment des lecteurs compatibles IBM, mais pas de lecteurs du type Apple.

Alors, tous à vos disquettes…

N.D.L.R. : ceci ne fonctionne pas sur les versions les plus anciennes du DISK MANAGER, versions qui ne posaient pas de questions relatives à la densité et au nombre de faces.

99

Petites annonces (gratuites)

A vendre modules : Basic Etendu (600 Francs), Mini-mémoire (600 Francs), Editor/assembler (600 Francs), Gestion de fichiers (250 Francs), Parsec (150 Francs), Multiplication 1 (70 Francs). Recherche documentation TI-Writer, LOGO II, Data Base Management, UCSD P-system.

Alain MACHUROT - 20, rue Raymond Bordier - 33200 Bordeaux - Tél. : 56.45.84.62 poste 1310 (heures de bureau).

Vends Mini-mémoire + Mini-assembleur + documentation Texas sur assembleur + livre d'initiation au langage assembleur du TI-99/4A : 650 Francs. 99 Magazine numéros 1 à 8 avec cassettes numéros 4 à 8 : 250 Francs.

JF. de Belleval - La Borne Trouée - 60153 Rethondes - Tél. : 44.85.62.26

Achète tous modules et lecteur, prix raisonnable. Possibilité échange (lecteur de disquettes) contre PB100 + interface, FX702P + interface + imprimante FP10, FX750P + interface + carte RAM 4Ko.

Michel Voisin - 22, route de Venerque - 31190 Auterive - Tél. : 61.08.21.88.

Vends TI-99/4A, boîtier d'extension. Cartes Pascal, RS232, 32Ko. Lecteur de disquettes. Modules Mini-mémoire, Éditeur/assembeur, Basic Étendu, trois modules de jeux. Documentation technique, livres, divers soft, manettes de jeux. Prix à débattre.

Yann Fasquel - 41, quai Eugène Lecorre - 78700 Conflans Ste Honorine - Tél. : 39.72.64.34 après 18 H.

Vends TI-99 + joysticks : 1000 Francs. Synthétiseur de paroles, Basic Étendu (Manuel en français) : 500 Francs pièce. Modules Adventure avec quatre cassettes : 400 Francs. Hopper, Burger Time, Star Trek, Return to Pirate's Isle, Super Demon Attack, TI Invaders, Mush, Tombstone City, The Attack, Treasure Island, Parsec : 100 Francs pièce. Cassettes Solar system, Lunar Lander : 80 Francs pièce.

Olivier Fontenay - 4, rue Armand, 78100 St Germain - Tél. : 39.73.33.42.

Vends TI-99/4A, interface Péritel, magnétophone Texas, cordons, manettes de jeux, Basic Étendu avec livres en français et anglais, mémoire 32Ko, TI Logo II, module Chasse aux Wumpus, Cassette Basic par soi-même, livre 50 programme pour TI-99/4A. Le tout : 3000 Francs.

Christian Delaunay - 3, bis rue du Colonel Oudot - 75012 Paris - Tél. : 43.41.97.18.

Vends TI-99/4A, Péritel, manettes de jeux, Basic Éten,du, deux modules (Foot et Musik), 16 livres, 30 cassettes, 99 Magazine 1 à 8. Le tout 2500 Francs. Vends aussi magnéto + câble, modulateur SECAM/UHF.

François Frattini - 30, galerie de l'Arlequin - Appartement 2224 - 38100 Grenoble - Tél. : 76.22.16.12.

Vends TI-99/4A (1984) Péritel, Ext. Basic français, câble K7, mémoire 32Ko externe, Mini-mémoire, interface Centronics, quatre modules de jeu, 25 cassettes de jeux et programmes, collection complète 99 Magazine, livres, manette Tirvit, 100 listings imprimés, trucs. Valeur 8500 Francs, vendu 5000 Francs.

Jean-Louis Philippe - 73, route de Sierck - Koenigsmacker - 57110 Yutz - Tél. : 82.50.01.46.

Vends TI-99/4A, Basic Étendu, synthétiseur de paroles, joysticks Texas, Magnétophone Philips, mini-mémoire + livre assembleur. Modules Beginning Grammar, Early Reading, Le Basic par soi-même, Aide à la programmation 1, Extended Basic par soi-même, Gestion Privée, Munchman, Football, Adventure, Poker, Tombstone, cassettes programmes de jeu, livres, série 99 Magazine.

Daniel Hervé - 98, rue du Lt Keiser - 95110 Sannois - Tél. : 34.10.14.12.

Vends Extended Basic (manuel en français) : 700 Francs. Cordon cassettes pour deux magnétophones : 100 Francs. Deux cassettes Basic par soi-même : 50 Francs chaque.

Frédéric Grenier - 6, rue Paul Signac - 78370 Plaisir - Tél. : 30.54.13.71 (après 17 H).

Vends TI-99/4A, adaptateur TV, manettes, magnéto avec câble, Basic Étendu, Mini-mémoire, TI-Calc, 22 modules éducatifs et jeux, 20 cassettes programmes divers, collection 99 Magazine, collection Hebdogiciel, Cassette Basic et Basic Étendu, Éditeur/assembleur. Valeur neuf 9800 Francs, vendu 4500 Francs (à débattre).

G. Lebon - Chemin des Montants - 54690 Eulmont - Tél. : 83.22.84.23.

Vends numéros 1 à 8 de 99 Magazine, 20 Francs pièce.

Sylvain Eichstadt - 6, rue de la Sapinière - 57600 Morsbach - Tél. : 87.85.42.46 après 14 H.

Le cadeau d'Atari®
à tous les possesseurs de VIC 20, Commodore 64, Apple II et TI 99/4A.



This collection is for all TI99 user. Greetings from Italy.
Astharot

Même si vous n'avez pas d'ordinateur Atari, Atarisoft vous permet désormais de découvrir les programmes Atari les plus célèbres : Pac-Man, Centipède, Jungle Hunt, Pôle Position, Galaxian, Miss Pac-Man, Joust, Moon Patrol, Dig-Dug, Donkey Kong, Robotron 2084, Defender, Stargate.

ATARISOFT™